IBM FileNet P8 Platform and Architecture

Architecture and expansion products

Solution design, creation, and implementation

Security, infrastructure, and scalability information

Wei-Dong (Jackie) Zhu
Nicholas Buchanan
Michael Oland
Thorsten Poggensee
Pablo E Romero
Chuck Snow
Margaret Worel

Redbooks

ibm.com/redbooks

International Technical Support Organization

IBM FileNet P8 Platform and Architecture

April 2011

SG24-7667-01

Note: Before using this information and the product it supports, read the information in "Notices" on page xi.

Second Edition (April 2011)

This edition applies to Version 5, Release 0 of IBM FileNet P8 Platform products.

Contents

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AIX®
Cognos®
DB2®
developerWorks®
Domino®
FileNet®
IBM®
ILOG®
Informix®
InfoSphere™
iNotes®
LanguageWare®
Lotus Notes®
Lotus®
Notes®
pureScale™
QuickPlace®
Quickr™
Redbooks®
Redbooks (logo) ®
Sametime®
Symphony™
System Storage®
Tivoli®
WebSphere®
z/OS®

The following terms are trademarks of other companies:

Adobe, the Adobe logo, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

SnapLock, NetApp, and the NetApp logo are trademarks or registered trademarks of NetApp, Inc. in the U.S. and other countries.

Java, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Preface

IBM® FileNet® P8 Platform is a next-generation, unified enterprise foundation for the integrated IBM FileNet P8 products. It combines the enterprise content management with comprehensive business process management and compliance capabilities. IBM FileNet P8 addresses the most demanding compliance, content, and process management needs for your entire organization. It is a key element in creating an agile, adaptable enterprise content management (ECM) environment necessary to support a dynamic organization that must respond quickly to change.

In this IBM Redbooks® publication, we provide an overview of IBM FileNet P8 and describe the core component architecture. We also introduce major expansion products that extend IBM FileNet P8 functionality in the areas of content ingestion, content accessing through connectors and federation, the application framework, and discovery and compliance.

IBM FileNet P8 Platform supports enterprise content management. In this book, we discuss the anatomy of an ECM infrastructure, content event processing, content life cycle, and business processes.

Each IBM FileNet P8 product has its own functionality, but they are all built on top of the IBM FileNet P8 Platform. The core platform provides the support for security around authentication and access control of processes and content of these products. In this book, we describe the security issues to consider in an enterprise environment, how IBM FileNet P8 addresses them, and how to manage security effectively in an IBM FileNet P8 environment.

Another important topic that we address in this book is how the unique ability of the IBM FileNet P8 Platform can scale horizontally and vertically to respond to increasing load demands. We discuss the available options for each of the core platform engines and for the expansion products.

This book gives IT architects, IT specialists, and IT Technical Sales a solid understanding of IBM FileNet P8 Platform, its architecture, its functions and extensibility, and its unlimited capabilities.

The team who wrote this book

This book was produced by a team of specialists from around the world working at the IBM Costa Mesa Software Development Lab, California, US.

Wei-Dong (Jackie) Zhu is an Enterprise Content Management Project Leader with the IBM International Technical Support Organization (ITSO). She has more than 10 years of software development experience in accounting, image workflow processing, and digital media distribution. Jackie has a Master of Science degree in Computer Science from the University of Southern California. Jackie joined IBM in 1996. She is a Certified Solution Designer for IBM Content Manager and has managed and led the production of many Enterprise Content Management, Risk and Discovery Redbooks publications.

Nicholas Buchanan is a Managing Consultant with IBM Enterprise Content Management Lab Services for Costa Mesa, IBM US. He has over twelve years of experience developing large scale information delivery solutions and brings extensive application development experience to IBM. Nicholas currently consults on IBM FileNet ECM architecture and solution design and has written extensively on storage, high availability, and business continuity.

Michael Oland is an IT Advisory Specialist with IBM Enterprise Content Management Lab Services in Washington, D.C., IBM US. He has 10 years of experience in enterprise content management, which includes web content delivery and management, document and information management, and business process management. For the last six years, he focused primarily on content migration and developing tools for content migration. Michael has a degree in Broadcasting from the University of Tennessee, Knoxville.

Thorsten Poggensee is a Managing Consultant working as an Enterprise Content Management Architect and Project Leader with Berlin, IBM Germany. He has more than nine years of experience in designing and implementing high-performance, high-volume, and high-availability solutions around the IBM FileNet product suite. Thorsten graduated as an Electrical Engineer from Hochschule fuer Technik, Wirtschaft and Kultur in Leipzig (Germany). He is a Certified Technical Support Provider for IBM Content Manager. Thorsten specializes in infrastructure implementation for IBM FileNet P8 solutions.

Pablo E Romero is a Certified Senior IT Specialist for the IBM Enterprise Content Management Center of Excellence based out of Research Triangle Park, North Carolina (NC), US. He has more than nine years of experience with enterprise content management technologies in various roles including development, quality assurance, and enterprise consulting. He has a degree in Computer Engineering from Florida Atlantic University. His areas of expertise are enterprise infrastructure and federation technologies. He has certifications as both an IBM Content Manager Solution Designer and a FileNet Technical Support Specialist.

Chuck Snow is an Enterprise Content Management Client Technical Specialist with the IBM Software Group. He has more than 10 years of experience selling and supporting complex enterprise solutions within the northeastern United States. Chuck has a Bachelor's degree from Temple University in International Business and an MBA from Clark University in Management Information Systems and Global Business. He is involved with many fields within information management, with specialization in records management and compliance solutions.

Margaret Worel is a Senior Systems Specialist Engineer (ITS), supporting demonstrations and rapid sales response for the IBM Enterprise Content Management Demo group in the US. She joined IBM through a merger with FileNet in 2006 and is currently focusing on demonstration best practices. Margaret is a graduate of the University of California, Los Angeles (UCLA) with over 12 years of software experience. She manages, designs, develops, and implements programs and procedures for data and process management.

Thanks to the following people for their contributions to this project:

Dan Whelan
Michael Seaman
Grace Chen
Patrick Doonan
Cheryl Greene
Lauren Mayes
Dana Morris
Joseph Raby
Grace Smith
Allen Takatsuka
Kristin Wallio
Mike Winter

Ben Antin
Kevin Bates
Roger Bacalzo
Mary Booher
Jerry Bower
Jay Brown
Jonathan F Brunn
Ku Chang
Jean Chen
Jingli Chen
Srinivas "Varma" Chitiveli
Dao-Quynh Dang
Eric Edeen
Mike Fannon
Kristoffer Gjevre
Frank Hayes
Philip W. Hirschi
Yong Jun
Terry Kemp
Darren Kennedy
Ross Lepiane
Wayne Malkin
Juergen Meutgens
Ute Werle-Muehlbach
Edward Nutt
Dave Perman
Jeffrey Peterman

Richard Robinson
Cengiz Satir
Sridhar Satuloori
Jenifer Schlotfeldt
Himanshu Shah
Darik Siegfried
Joerg Stolzenberg
Steve Studer
Firas Yasin
Shuang Wang
Shawn Waters
Marcus Mueller-Westerholt
Charles F Wiecha
Sean Yang
Margaret Yu

Thanks to the authors of the previous editions of this book.

- Authors of the first edition, IBM FileNet P8 Platform and Architecture published in July 2009, were:

  Wei-Dong Zhu
  Kameron Cole
  Adam Fowler
  Michael Kirchner
  Bruce J Mcdowell
  Chuck Snow
  Mike Winter
  Margaret Worel

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

- Follow us on Twitter:

  http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

Summary of changes

This section describes the technical changes made in this edition of the book and in previous editions. This edition might also include minor corrections and editorial changes that are not identified.

Summary of Changes
for SG24-7667-01
for IBM FileNet P8 Platform and Architecture
as created or updated on April 20, 2011.

First edition (July 2009) covers IBM FileNet Platform Version 4.x

April 2011, Second Edition

This revision reflects the addition, deletion, or modification of new and changed information described below.

New information

- Content Federation Services
- Content Search Engine
- Other new features of IBM FileNet P8 Version 5.0.

Changed information

- All chapters have been revised with the latest features and functions of FileNet P8 Version 5.0.

For what's new in IBM FileNet P8 Version 5.0, refer 1.4, "What is new in 5.0" on page 15.

1

IBM FileNet P8 Platform overview

IBM FileNet P8 Platform is a unified enterprise foundation for the integrated IBM FileNet P8 products. The latest version expands its capabilities to address vital business requirements with innovative solutions. It combines enterprise content management, comprehensive business process management, and extensive compliance capabilities to address a wide range of content-related business requirements. IBM FileNet P8 Platform is the key element in creating the adaptable enterprise content management environment necessary to support a dynamic organization that must respond quickly to change. This chapter describes the issues that surround the exponential growth of electronic content and how IBM FileNet P8 Platform and its core products address these challenges. New advantages and capabilities in the 5.0 release are highlighted. It also introduces IBM FileNet P8 products that take advantage of these enterprise capabilities to further expand on the value proposition that IBM FileNet P8 has to offer.

The chapter covers the following topics:

- 1.1, "Information explosion" on page 2
- 1.2, "IBM FileNet P8 Platform overview" on page 5
- 1.3, "Advanced topics" on page 14
- 1.4, "What is new in 5.0" on page 15
- 1.5, "Summary" on page 17

1.1 Information explosion

Businesses today face an information explosion and struggle with growing stores of unmanageable and unsearchable documents. Corporations create increasing volumes of both paper and digital content. The sheer volume of this information poses significant obstacles to productivity, escalates storage costs, and exposes an organization to regulatory, competitive and operational risk. They need a solution to make this information available, active, consistent, and reliable. They need an enterprise content management system.

Companies need to take advantage of content as a valuable business asset. This system needs to address corporate requirements to control and access content. Additionally it needs to take that content and make it an active part of business processes. It needs to provide a structure to meet compliance requirements, and enhance the information to make it more searchable and extract business trends.

Controlling costs while implementing these goals are difficult, as users have high expectation for content access. They demand trusted, quick, and easy content delivery. Harnessing this information improves business efficiency, delivers customer value and addresses regulatory compliance. Organizations that effectively manage their content explosion increase productivity and deliver consistent customer value.

The IBM FileNet P8 is an enterprise level platform with modular solutions that manage content, optimize business processes, and enable compliance and smarter archiving. The IBM FileNet P8 Platform is agile and highly scalable. It enables organizational leaders to capitalize on this information by aligning its use to business directives. By implementing Enterprise Content Management strategies these business benefits can be achieved in a cost effective way.

This chapter overviews the IBM FileNet P8 Platform and how it manages content and turns information into a vital business asset. It introduces subjects covered in subsequent chapters, and their relationships to the FileNet P8. In addition to the core platform products and add-on modules which enhance the platform's capabilities are described. Following this section is a more detailed discussion of the issues faced in content management finally mapping those issues to FileNet P8 solutions.

1.1.1 Content management goals

Harnessing content as a strategic asset is an organizational challenge. Content management requirements cover four main areas: content management, process, compliance, and discovery. This section discusses these areas in more detail.

Content

What needs to be managed? Finding all the locations that contain documents and monitoring those repositories can be a major challenge. In addition to the variety of locations and formats, the volume of content is a main factor. These documents must be located and cataloged to be managed effectively.

Is this the correct version? Business applications often require access to data that resides in multiple, distinct repositories that were never intended to share their content with other systems. One common approach to try to address these silos is to simply duplicate electronic content for each system that needs to consume it. Duplicate copies, outdated information, and non-business related files clutter storage systems. This introduces yet another set of information efficiency, storage, and maintenance issues.

Who can use it? Additionally, these assets must be made secure.As a document goes through its life cycle, the solution must ensure the correct people can author and consume content needs and continue to control this during each phase. Restricting access to confidential, proprietary, and other sensitive content is greatly complicated when files are spread across many systems and storage environments that lack a central management facility.

What if there are documents that must be worked on collaboratively? How can someone create a process to work together on demand without hard coded, deterministic processes?

Process

How will the content be used and accessed? Corporations want to invest in platforms that have low maintenance, longevity, stability, and most importantly, ease of use. The user interfaces must be clear and require little training. It must support the tools of their business.

How can information be integrated? Users need a complete view from all information sources, using the correct data, to make correct decisions, which liberates clients to take advantage of existing knowledge to innovate rather than re-create.

How can processes be flexible and agile? Business need to meet constantly changing market conditions, and yet keep costs down. A content management system must implement changes easily, without requiring a large and specialized development department.

How can process be innovative? Managers have the business knowledge to change process. A content management system must enable them to make these changes without needing to transfer that knowledge to another team.

Removing the obstacle of time and communication errors accelerates business responsiveness.

Compliance and discovery

How does someone know if they have all of the information they need to make a decision? Users need a complete view from all information sources, using the correct data, to make correct decisions. They need to know the current conditions and context.

How can businesses control, archive, and dispose of documents accurately? Organizations need to maintain corporate records control policies without slowing down business. User behavior and consistency can increase the risk of non compliance and prevent the possibility of loss of critical documents.

How do corporations protect themselves from loss of important information? How do they minimize risk in legal processes? Audits and legal discovery efforts involve identifying and producing numerous documents, e-mails, and other content. Organizations need to respond swiftly, efficiently, and correctly to these requirements.

How can businesses find trends and meet auditing needs? Audits, compliance, and legal discovery efforts involve identifying and producing numerous documents, e-mails, and other content. It is also valuable to mine data out of this information, as well as detecting new developments.

How can businesses reduce storage costs and meet legal requirements? Compliance solutions also need dispose of documents in an appropriate fashion. Storing documents past their usefulness is a large infrastructure expense. Additionally, disposing of documents is often needed to meeting privacy, industry, and legal rules.

1.1.2 Enterprise content management solution requirements

All of these requirements and more can be stated in a few overarching needs. An enterprise content management systems must meet these needs by providing information that is timely, accurate, and secure. It must also enhance knowledge assets by activating it with processes and business intelligence.

Content

The system must control and manage all forms of content. It must reduce costs that are associated with storing the content. It must make the information available securely and efficiently at every phase of its life cycle.

Process

The system must support process modeling and workflows. It must intelligently handle not only predicable work but also workflows that must address new or unique solutions on demand.

It must allow the business to monitor work in progress and enable historical analysis to drive process improvement.

Compliance and discovery

The system must:

- Enable analysis and insight into the knowledge stored.
- Index and classify analysis and insight so that it can be found intelligently
- Automatically keep important information and dispose of it at the end of the life cycle
- Provide investigation tools and reports to support auditing and business intelligence

Modern enterprise management systems expand on traditional content life cycle management capabilities, such as workflows, content analysis and classification, and business intelligence that raise solutions beyond file shares. Enterprise content management systems become an integral part of corporate operation and decision making.

1.2 IBM FileNet P8 Platform overview

The IBM FileNet P8 Platform is a unified-enterprise content management and business process management platform that supports all of these requirements. It resolves content, process, compliance, and discovery issues in one single solution. The platform's tight integration increases operational efficiency by reducing the number of products and vendors across the enterprise.

The platform architecture provides interoperability across standard organization applications, file stores, and data repositories, which reduces integration costs, increases accuracy, and accelerates value. The integrated approach effectively addresses the complex demands of managing content across an enterprise.

FileNet P8 enables new capabilities to be rapidly developed with proven workflow capabilities and case management functionality. Organizations can respond efficiently to new market requirements with agile development applications. Classification, analysis, reporting, and discovery tools give transparency to business activities. Using these tools, management can optimize and accelerate productivity.

1.2.1 FileNet P8 Platform ECM solutions

The IBM FileNet P8 Platform enables enterprise solutions by providing a breadth and depth of core functionality. FileNet Content Manager provides content, security, and storage. FileNet Business Process Manager supplies workflows, decision making, and productivity. These two core solutions facilitate resolving critical organization needs and optimizing business outcomes.

FileNet P8 helps organizations optimize processes, shorten production times, and improve productivity and accuracy. It includes process design and simulation tools, electronic forms, application development frameworks, and monitoring and reporting tools. It also includes user interface designed for the business user. FileNet Content Manager also supports collaboration, and widgets, integration with Microsoft® Office 2007 and more.

The P8 Platform also supports rapid application development, leveraging Web 2.0 technology in user-oriented application environments. This accelerates creation and deployment of case-based applications across the enterprise. FileNet comes with pre-wired ECM Widgets in a Web 2.0 Mashup-based framework that accelerates creating solutions and reduces dependence on IT resources. By using open standards, corporations harness maximum flexibility and interoperability with popular third-party applications and frameworks.

Content

FileNet Content Manager controls and manages all forms of content. It includes document versioning, content security, and life cycle management. These key features enable *content anywhere*, so that corporations can access that same content from within customized and industry-standard applications. They can choose to manage content in-place, postpone or avoid costly content migrations, yet still standardize on P8. Solutions can preserve existing assets or consolidate on less costly storage.

Process

FileNet P8 activates content by making it a participant in business activities. *Active content* means that actions that are associated with content can trigger predefined actions and enable automation. The P8 Platform contains processes modeling tools, case type workflows, and supporting nonstandard processes. Monitoring tools give insight into a work in progress, giving transparency to key business matrix. Modeling and analytics capabilities support effective process design before implementation and process improvement after deployment.

Compliance and discovery

The P8 Platform has both included and expansion components that cover a wide range of compliance tools and solutions. This tool set includes federation and collection, classification, analysis, and records management tools. This suite of programs is part of Information Lifecycle Governance, which ensures that content is trustworthy from acquisition to disposition. A subset of tools in the suite, eDiscovery Search and Analytics, solves investigation and auditing issues with case assessment and management.

This functionality is supported by the IBM FileNet P8 Platform architecture. The next section covers how the architecture accomplishes these solutions and expands on them.

1.2.2 FileNet Enterprise Reference Architecture

The IBM FileNet P8 Platform has a unified Enterprise Reference Architecture (ERA) that is a blueprint for designing content and business process management solutions. The reference architecture is a visual depiction of the P8 Platform, divided into several building blocks, or layers, that are centered around functional responsibilities, which are used to communicate the technological vision and architectural approach of the platform. This reference architecture was developed to be extensible using a pluggable component based system, which emphasizes, abstraction, interoperability, and agility, as shown in Figure 1-1.

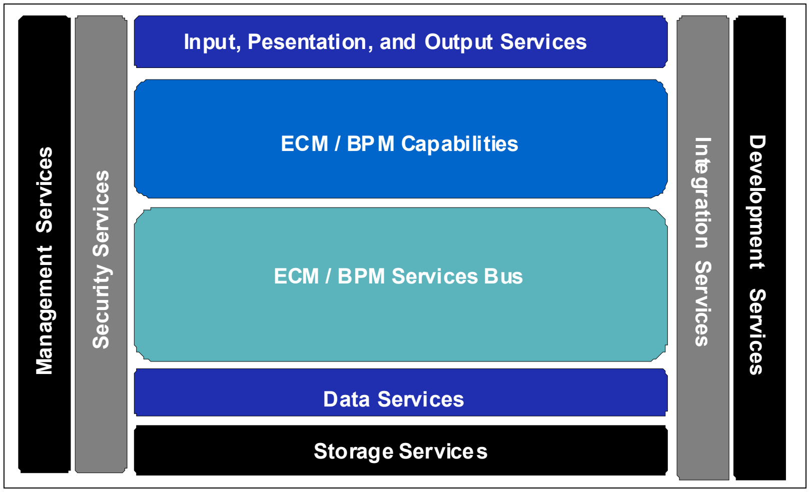


Figure 1-1 High level Enterprise Reference Architecture

At the highest level, there are seven layers grouped in two areas:

- Central capabilities and services:
  - Input, presentation, and output services: User interfaces, application building blocks, capture and acquisition of content, and output
  - ECM/BPM capabilities and services: Core of P8 Platform, providing content management and business process functionality
  - Information infrastructure: Data and storage services
- Vertical layers:
  - Management services: Process and system management
  - Security services: Access control and management
  - Development services: Expedites application creation
  - Integration services: Integration with data systems, repositories, and other applications

Central capabilities and services

Three of these layers, input, presentation, and output services, ECM/BPM capabilities and services, and the information infrastructure are the center of the framework. Starting from the bottom up, they support each other in forming the innermost structure that supports the solution architecture.

Information infrastructure

The foundation of the architecture contains data services and storage services. Data services include integrity, encryption, and other functionality related to data integrity. The data service layer abstracts calls to data stores and applications. Storage services provide an abstraction between FileNet and a variety of storage, data, and content cache services.

ECM/BPM capabilities and services

Building on the information infrastructure, Content and Process Management Services provide content and process management modules. These services are provided by core engines discussed later in this chapter and in detail in following chapters.

Input, presentation, and output services

Input and Presentation Services collect, capture, and archive assets from a variety of sources into the ECM system. This layer also provides display and interface supports for users and applications.

Vertical layers

Several key capabilities complete the ECM architecture. Management, Development, Integration, and Security Services apply to all layers. These services provide critical functionality outside of enterprise content management operations.

Management services

Management Services include system and process management and also give insight into the health of the system through monitoring tools. High availability, disaster recovery, and capacity planning functions are located here as well as cloud support.

Security services

Security Services include modules for identity management, perimeter, and security models.

Integration services

The Integration Services layer includes APIs, SOA, and orchestration functions. Tools, such as Master Data Management, Business Intelligence, and collaboration offerings, increase insight and productivity in business operations.

Development services

The Development Services layer have located process, application, form, object, and design and development tools. This rich tool set is accompanied by test and debugging features and application deployment structures.

1.2.3 IBM FileNet Platform core components

All of the layers that we previously mentioned are formed around the core ECM/BPM Capabilities and Services, which are in turn based on three core components, as shown in Figure 1-2 on page 10:

- Content Engine: Provides the core content management capability, including versioning, security, and life cycle management
- Process Engine: Provides the core workflow capability (business process management) including modeling, electronic forms, and monitoring tools
- Workplace/Workplace XT: Provide user interface to access content and process engines

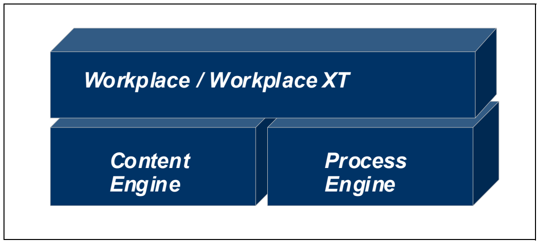


Figure 1-2 Core FileNet P8 Platform core components

The Workplace/Workplace XT, Content Engine, and Process Engine are foundational components of the ECM reference architecture. This reference architecture is extensible with a pluggable-component based system that enables abstraction, interoperability, and agility. This section provides a high-level overview of the components and the functions that they perform in an ECM solution.

Application Engine and Workplace naming convention: Application Engine is the official name for Workplace. Application Engine does not equate to Workplace XT. Both Workplace and Workplace XT support a common set of functions but differ in other areas. For consistency of the terminology used in the book, we use Workplace instead of Application Engine throughout the book.

1.2.4 Application interfaces

P8 Platform provides several options for application interfaces.

Workplace and Workplace XT

Workplace and Workplace XT are web interfaces that expose both content management and business processes in an intuitive full-featured application. Workplace and Workplace XT are two different applications, but both access the process engine and content engine with slightly varying functionality. Workplace is customizable while Workplace XT is not.

IBM FileNet integration for Microsoft Office

A component of IBM FileNet Content Manager and IBM Business Process Manager, IBM FileNet Integration for Microsoft Office is a tightly integrated interface exposing tasks, processes, and content management.

IBM FileNet eForms

A component of IBM FileNet Business Process Manager, IBM FileNet eForms provides an electronic alternative to paper forms. With an easy to use visual design environment, eForms creates highly intelligent forms without requiring coding or scripting.

IBM FileNet Business Process Framework

A component of IBM FileNet Business Process Manager, IBM FileNet Business Process Framework is a flexible foundation that accelerates application development and time-to-market while reducing development effort. It includes case-based functionality, role management, and work distribution. Workplace works with Business Process Framework. Workplace XT does not.

ECM widgets

FileNet P8 Platform streamlines development with out-of-the-box, pre-wired widgets enabling drag-and-drop applications. With widget support, the P8 Platform enables business analysts to easily create rich composite applications.

1.2.5 Expansion products

This book covers a variety of extension products that implement important requirements for content management systems. In particular, these offerings fall into two main categories, applications that bring content into FileNet and applications that connect and federate documents into FileNet:

- Expansion products for content ingestion:
  - IBM Content Collector
  - IBM Datacap
  - IBM FileNet Capture
- Expansion products for connection and federation:
  - IBM FileNet Services for Lotus Quickr
  - Content Management Interoperability Services
  - IBM FileNet Content Integration Services
  - IBM FileNet Content Federation Services
- Expansion products for Information Lifecycle Governance:
  - IBM Enterprise Records
  - IBM Classification Module
  - IBM Content Analytics
  - IBM Collection Collector
  - IBM eDiscovery Manager and eDiscovery Analytics

Here is a brief introduction of these products. We cover them in more depth in the subsequent chapters.

> **Note:** The functionalities of IBM FileNet Connectors for Microsoft SharePoint and IBM FileNet and IBM FileNet Application Connector for SAP are rolled into the following products respectively:
>
> - IBM Content Collector for Microsoft SharePoint
> - IBM Content Collector for SAP Applications

Expansion products for content ingestion

Content ingestion products in FileNet do more than just copy, move, or link documents. They also index, de-duplicate, classify, manipulate, activate, control, and perform a number of other actions that add value and harness mission critical information. This set of products includes IBM Content Collector, IBM Datacap, and IBM FileNet Capture.

IBM Content Collector

IBM Content Collector, formerly called IBM InfoSphere™ Content Collector, provides powerful tools for collecting and managing all types of content, regardless of storage location. Content Collector is available in three offerings:

- Content Collector for File Systems
- Content Collector for Email
- Content Collector for Microsoft SharePoint.

IBM Datacap

The IBM Datacap product line features a complete suite of on and offline capture and image processing solutions. These products offer comprehensive document refinement tools and is a full-function capture platform for forms processing and document identification.

IBM FileNet Capture

IBM FileNet Capture provides the ability to store images of paper and fax documents in P8. It automatically extracts textual data from free form, semi-structured and structured documents.

Expansion products for connection and federation

Often there are valid reasons for leaving content in their current locations. Organizations might want to preserve existing investments or prefer to work in environments with particular interfaces or functionality. The following set of products bring the power of the FileNet P8 Platform to these solutions, expanding utility and preserving assets. This set includes IBM FileNet Services for Lotus® Quickr™, Content Management Interoperability Services, IBM FileNet Content

Integration Services, IBM Content Federation Services, and support for the Content Management Interoperability Standard.

IBM FileNet Services for Lotus Quickr

IBM FileNet Services for Lotus Quickr combines the teamwork features of Lotus Quickr with the content and business process management of IBM FileNet. Lotus Quickr provides a web user interface, team places, and integration with desktop, e-mail, and document creation applications in a collaborative environment.

Content Management Interoperability Services

The Content Management Interoperability Services (CMIS) standard is an interface adopted across the content management industry. It is used to rapidly develop applications that are reusable across a variety of content repositories without custom development.

IBM FileNet Content Integration Services

IBM Content Integrator (previously known as IBM Information Integrator Content Edition) incorporates content in existing repositories and makes that information available to the IBM FileNet P8 Content Manager though the Content Federation Services module. CFS connectors, built by customers and partners, use IBM Content Integrator's well-established connector architecture.

IBM FileNet Content Federation Services

A component of IBM FileNet Content Manager, IBM FileNet Content Federation Services provides a unified view of unstructured content by providing a single point-of-access to content across multiple repositories.

Expansion products for Information Lifecycle Governance

The Information Lifecycle Governance suite includes content assessment, collection and archiving, classification, records management, and discovery components. They focus on records control, legal compliance, responding to discovery requests, and creating value in business information by making it trustworthy and relevant.

IBM Collection Collector

See the overview in “IBM Content Collector” on page 12.

IBM Enterprise Records

IBM Enterprise Records, formerly IBM FileNet Records Manager, classifies, applies holds and retention policies, and stores electronic records. Using Content Federation Services, Enterprise Records can apply record controls to documents and content that reside in other repositories as well. IBM Enterprise records provide an organization with the ability to treat content as legal business records and apply retention and disposition rules, which can be used to comply with IT, industry, and legal requirements.

For more information, refer to the product manuals and the following IBM Redbooks publication: *Understanding IBM FileNet Records*, SG24-7667.

IBM Classification Module

IBM Classification Module combines text analysis classification to categorize more accurately and efficiently, making information more organized and useful.

For more information about IBM Classification Module, refer to the product manuals and the following IBM Redbooks publication: *IBM Classification Module*, SG24-7707.

IBM Content Analytics

IBM Content Analytics provide analysis capabilities that aggregate, correlate, explore, and display information in a contextually relevant manner. These capabilities enable an organization to retain and manage only what is needed and decommission or delete the rest.

For more information about IBM Classification Module, refer to the product manuals and the following IBM Redbooks publication: *IBM Classification Module*, SG24-7877.

IBM eDiscovery Manager and eDiscovery Analytics

IBM eDiscovery Analytics and IBM eDiscovery Manager focus on the needs of collecting and processing unstructured data for legal discovery.

IBM Content Analytics formerly called Cognos® Content Analytics, replaces IBM Content Analyzer. IBM Content Analytics helps organizations discover, refine, visualize, and deliver new business insights through the analysis of unstructured content.

1.3 Advanced topics

Additionally, this book covers a number of topics that are relevant to the P8 Platform, which includes advanced case management with IBM Case Manager, Security, Scalability, and Solution Architecture. ACM is a new paradigm for solving customer issues from a content, rather than a process, standpoint. This case management solution frees knowledge workers to solve problems appropriately for the current situation.

Security issues are always an important topic, and the FileNet platform has fine- grained support for a wide variety of solutions. Likewise, load and scalability are critical components of enterprise solutions and many critical topics are addressed here.

Finally, this book also discusses how to architect solutions and solve business problems. Chapter 10, “Architecting an IBM FileNet P8 solution” on page 333 discusses how the various parts of the platform are evaluated and designed to form a complete end to end solution.

1.4 What is new in 5.0

The 5.0 release of IBM FileNet P8 includes many important new features:

- Content Federation Services
- Full text indexing and searching engine change
- Fixed content device support addition
- Usability and functionality improvements
- Operating system support

Content Federation Services

Content Federation Services delivers moving content for CS, OTEX, DCTM, CM8, and custom built connectors. Federated content from these repositories can be moved to non-federated storage areas, such as a fixed content device or a file system.

Full text indexing and searching engine change

IBM will begin to phase out the current implementation of the Content Search Engine (based on Autonomy K2) and replace it with Content Search Services based on Lucene. Both engines are available and can be run concurrently in this release. Indexes in 5.0 are created on a per object store basis to improve performance and scalability.

Fixed content device support addition

Content Engine 5.0 adds support for Hitachi Content Archive Platform and continues to support IBM N-Series SnapLock®, IBM Information Archive, NetApp® SnapLock, and EMC Centera.

Usability and functionality improvements

Content Engine 5.0 improves database support for DB2®, expanding property support and additionally allowing for page sizes in excess of 32K. Content Engine 5.0 LDAP now allows administrators to specify the LDAP attribute to be used as a security identifier. This release also features a new web management interface for configuring the Content Engine in a new system health dashboard. The client installers were changed to dynamically download their libraries from the server.

The Process Engine was rewritten to be a pure Java™ application and no longer requires root privileges to install. Much of the configuration for Process Engine was moved from the installer to the Process Engine run time. Process Engine now supports running multiple instances in the same operating system, allowing multiple independent installations to coexist. Each process engine instance runs against an independent database called a process store. Each process store supports multiple isolated regions. The process of creating isolated regions was also simplified. IBM FileNet BPM improves IBM Content Manager 8 support and integrates directly with IBM Cognos 8 Business Intelligence and IBM Cognos Real Time Monitor. The Process Engine now has Chinese National Standard GB 18030 support, allowing the use of both simplified and traditional Chinese characters. Also supported is UI mirroring for Arabic and Hebrew languages.

Operating system support

The IBM FileNet P8 5.0 release supports running Process Engine on Linux® and zLinux. Beginning with this release P8 5.x, all of the core engines support Linux environment.

Standards

IBM FileNet P8 5.0 supports the OASIS CMIS standard. CMIS is a standard for client/server APIs aimed to improve interoperability across ECM repositories. The 5.0 release also supports the Federal Information Processing Standard (FIPS). See 5.3, “Content Management Interoperability Services” on page 125 for a more detailed description of how to exploit this standard.

Deprecated APIs

IBM FileNet Content Manager 5.0 will sunset support for 3.5 API compatibility. The following is a list of deprecated functionality:

- The Direct Internet Message Encapsulation (DIME): Superseded by Message Transmission Optimization Mechanism (MTOM). developers are encouraged to use the MTOM URI (Uniform Resource Identifier) instead.
- Java Compatibility API/Buzz API for 3.0 and 3.5 i: Developers are encouraged to begin using the 4.x APIs.
- The Component Object Model (COM) compatibility API for 3.0 and 3.5: Developers are encouraged to begin using the 4.x .Net APIs.
- Content Engine Web Services API for 3.5: CEWS 4.0 API supports all the elements and functions of CEWS 3.5.

1.5 Summary

This chapter discussed the problem of content explosion and the resulting need for a platform-based approach to resolve those issues. Also discussed were the capabilities and high-level components of the IBM FileNet P8 Platform. The complete IBM FileNet P8 operating environment is presented throughout this book with the aim of demonstrating how IBM FileNet P8 can be leveraged to drive real content- and process-driven solutions.

2

Core component architecture

IBM FileNet P8 Platform is the unified enterprise foundation for the integrated IBM FileNet P8 products. This chapter describes the core components of IBM FileNet P8 Platform, their architecture, data model, and associated security features.

This chapter covers the following topics:

- 2.1, “Core components overview” on page 20
- 2.2, “Content Engine” on page 21
- 2.3, “Process Engine” on page 40
- 2.4, “Workplace and Workplace XT” on page 51
- 2.5, “Configuration tools” on page 53
- 2.6, “Case Manager” on page 55

2.1 Core components overview

IBM FileNet P8 Platform is a collection of tightly integrated components that are bundled together under a common platform. The broad functionality provided by these integrated components constitute an enterprise content and process management platform. Some of the key elements of this platform are a metadata repository, a process management repository, an out-of-the-box user interface for accessing content and process elements, and a storage framework that can support a wide range of storage devices and platforms.

To provide these services, the IBM FileNet P8 Platform relies on three core components:

- Content Engine (CE)
- Process Engine (PE)
- Workplace (WP) / Workplace XT (WP XT)

All other add-on products are built on the foundation that these three components provide.

Figure 2-1 provides a high-level architectural view of these key components and their relative interactions.

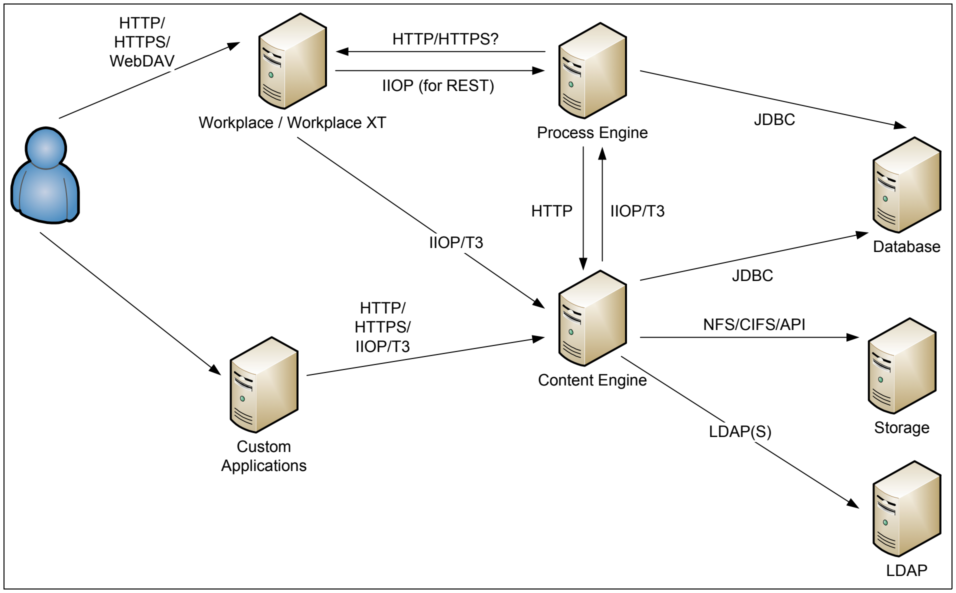


Figure 2-1 IBM FileNet P8 core components

Application Engine and Workplace naming convention: Application Engine is the official name for Workplace. Application Engine does not equate to Workplace XT. Both Workplace and Workplace XT support a common set of functions but differ in other areas. For consistency of the terminology used in the book, we use Workplace instead of Application Engine throughout the book.

The primary components are delivered as stateless components with farmable architectures that can support both vertical and horizontal scaling. Technology variations in the servers affect the way in which each is load-balanced and provisioned. Regardless, each server provides highly performing and highly scalable services in support of handling even the largest enterprise's mission-critical loads.

The remaining sections of this chapter provide details about each of the main components: Content Engine (including the storage tier), Process Engine, and Workplace/Workplace XT. In addition, there is a section that reviews the new Case Manager. For a review of additional included user interfaces, including Workplace XT, see Chapter 3, “Application interfaces” on page 63.

2.2 Content Engine

The Content Engine stores and retrieves all content within an IBM FileNet P8 system. It provides a series of services for creating, retrieving, updating, deleting, and securing content. In addition, it provides interfaces for handling event-based actions, document life cycle, and integration with various storage mediums.

The Content Engine is a J2EE Enterprise Java Bean (EJB) application and is deployed to a supported J2EE application server. The IBM FileNet P8 Platform currently supports a variety of Java application servers, such as IBM WebSphere®. For the complete list of supported application servers, see the Hardware and Software requirements guide.

The Content Engine application is written based on a generic set of J2EE services and is implemented primarily as a set of stateless session beans that take advantage of JDBC to store the metadata in the underlying database. The Content Engine also leverages both native and API-based interfaces to store content in a variety of storage medium. These J2EE services are fronted by two stateless EJBs that make up the EJB Listener, which demarcates the transaction and authentication boundaries into the server. Various asynchronous activities within the server are managed by a series of background threads.

Figure 2-2 shows the internal system architecture of a Content Engine.

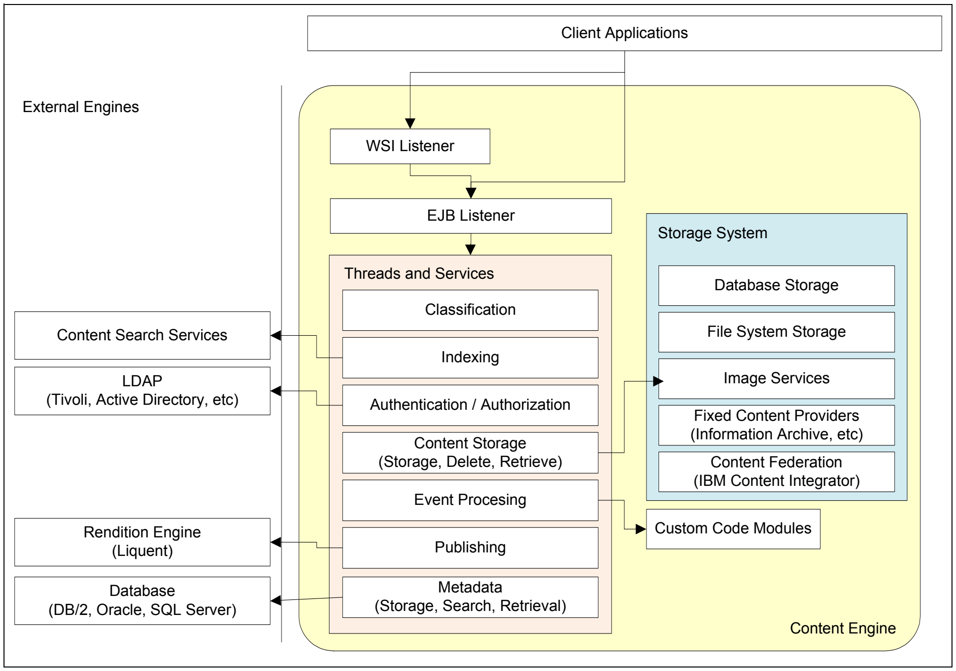


Figure 2-2 Content Engine internal system architecture

Organization

Each Content Engine server instance is configured as a member of a given IBM FileNet P8 domain. A Content Engine server instance can support one or more object stores where content is stored. Each object store can be scaled to store hundreds of millions of objects and to service requests from thousands of concurrent users.

There is a single global configuration database (GCD) per IBM FileNet P8 domain. The GCD contains the system configuration, marking sets, and registration for all object stores that are defined in the system. Object Stores define a set of content, properties, and storage areas that represents a data set.

Figure 2-3 on page 23 illustrates the Content Engine's internal data structure.

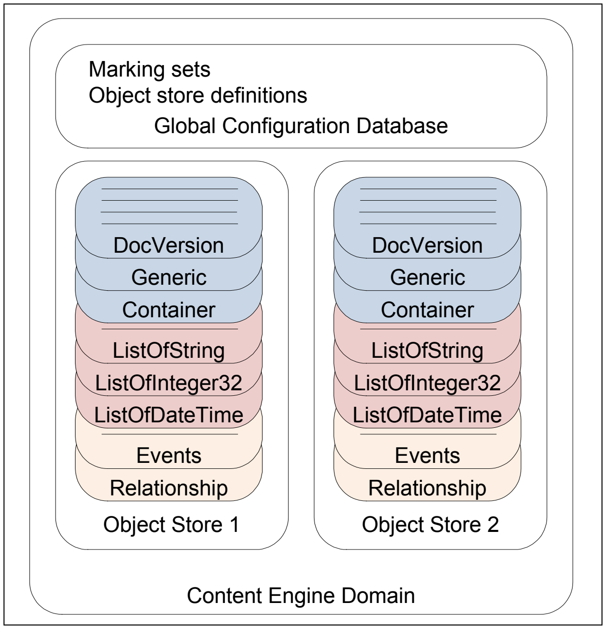


Figure 2-3 Content Engine internal data structure

2.2.1 Data model

The Content Engine stores documents and metadata in a combination of a database and target storage. The Content Engine uses an object-oriented data model defined by properties, classes, custom objects, and folders.

Central to the Content Engine is a strongly typed, hierarchical, and extensible object model. The Content Engine comes with a set of defined system classes that you can extend to create custom classes for use within an application. Some examples of the predefined classes are Document, Folder, Custom Object (special object that is content-free with metadata information only), Annotation, and Referential Containment Relationship.

Metadata and properties

Object information and metadata are defined and stored as properties for a given Content Engine object. These properties can be defined as one of the following property types: string, 32-bit integer, 64-bit floating point, binary data, ID (a GUID), date and time, boolean, and another object (object-valued property). The property can be defined as either containing a single value or multiple values (single- or multi-valued). In addition, it is also possible to configure a property's default value(s) and to define whether a property value is required, how the values persists across versions, and when the value can be set (on creation only or at will).

Object-valued properties operate differently than the other data types. Single valued properties do not contain a simple value (for example a single string or a number), but instead contain a pointer to another object in the system, for example, a Folder object has an object-valued property named Parent referencing the Folder that it is contained in. Unlike other data types, a multi-valued object property does not contain an actual list of other objects, but instead it works as a reflective object property, meaning the values of that property are other objects that are pointing back to the object. Figure 2-4 illustrates the difference between single-value and reflective object property.

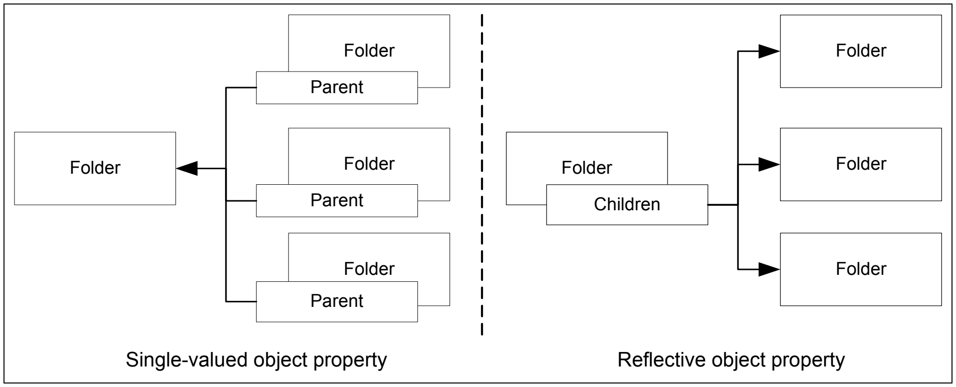


Figure 2-4 Single and reflective object properties

The cardinality of the property (single or multi-valued) defines whether the values are stored with the object record or in a series of multi-value tables (ListOfString, for example).

Classes

Within the Content Engine, each object is defined as an instance of a *class*. The class defines what type of object is being created, what metadata is to be collected about that object, and how that object is to be stored in the system. In keeping with the object-oriented nature of the Content Engine, classes can be subclassed. This orientation allows for a parent class to define a set of shared properties and behaviors; therefore, the subclass can define additional properties and possibly override the parent's behavior as necessary.

Documents

The Content Engine allows for storing content as document objects. Document objects have metadata (properties) and one or more content elements. In addition, documents can optionally have life cycle and versions. For versioning, a document can be versioned at both a major and a minor level. Check-out and check-in operators provide facilities for locking a given revision for a given user.

Each independent document object results in a single row in the DocVersion table and are linked together by a *VersionSeries* object.

Custom objects

Custom objects are metadata-only objects with no associated content elements and are not versionable. Custom objects are typically thought of as a collection of metadata and possibly links to other objects within the IBM FileNet P8 repository.

Each object creates a new record in the Generic table.

Foldering

Content Engine offers a folder and foldering mechanism for organizing content by related items into a file system-like structure. Every folder in the system exists in the folder tree that derives from the root folder. Any document or custom object can be filed into one or more folders using a *referential containment relationship* object.

Because a folder is an object, it too can have metadata properties assigned to it. Each folder has a set of default metadata that is assigned (date created, creator, parent folder, and name). The folder class can also be subclassed to allow for additional metadata to be collected and stored.

Each folder results in a single row in the Container table.

2.2.2 Security

In addition to storing the content and metadata in the repository, the Content Engine offers a robust security model to allow for authenticating users and then controlling whether a user has rights (or is authorized) to use, view, update, and delete any given object or resource in the repository.

Authentication

Authentication is the process of determining who users are and whether users are who they say that they are. For authentication, Content Engine relies on the J2EE authentication model, which is based on the Java Authentication and Authorization Service (JAAS). Content Engine relies on the application server to perform this JAAS authentication. This leverages the security of the application server itself, which might do the authentication itself or might use perimeter authentication, if the authentication was already performed by another front-end system.

Content Engine (and IBM FileNet P8 in general) does not contain its own user database. Instead it leverages the existing user repository within an organization. This repository is accessed using LDAP and eliminates the need to maintain a user sync between the corporate user repository and the IBM FileNet P8 system.

Authorization

Authorization is the process of determining whether a user is allowed or disallowed permission to perform an action on an object, such as view or update. Content Engine manages these permissions right through an access control-based authorization model. Individual access rights control which actions can be performed on a given object by a given user or group (known as *principals*). These individual access rights, called *Access Control Entries or ACEs*, can be grouped together to form an *access control list*, or ACL. Applying one or more ACLs to an object individualizes the level of security that is enforced on an object. ACLs can apply directly to the object, or the object can be secured by ACLs derived from other sources:

- Default: Each class is created with a Default Instance Permissions ACL. By default, the Default Instance Permissions ACL is applied to instances of the class.
- Templates: Security templates, which contain a predefined list of access rights, can be applied to an object.
- Inherited: Permissions applied to some objects (folders, documents, or custom objects) can be inherited by other objects so that these “inherited” permissions supplement other permissions on the receiving objects. See “Security Inheritance” on page 27 for more information.

Principals that are defined within an ACE for a given ACL represent users or groups that are defined within an underlying LDAP repository. The task of mapping an access right to a user or group is supported by the authorization framework, which manages the user and group look-up in the configured LDAP repository or repositories. Content Engine supports most of the widely used LDAP stores (including Tivoli® Directory Server, Active Directory, and SunOne Directory Server). For more information, refer to IBM FileNet P8 Hardware and Software Requirements.

When generating an ACL, the Content Engine stores in the ACE a unique identifier for the user or group. In previous versions, which unique ID to store was not configurable, but in 5.0, the desired LDAP attribute to use as the unique ID can be specified during initial configuration, which allows more flexibility when moving systems between providers or within an organization.

For more information about security, see section 8.3.1, “Security of Content Engine objects” on page 215.

Security templates

Security templates are used to apply a specific set of rights to a document based on the values of the template. Templates can be combined with a document life cycle to easily change the security of the document based on where it is in its life cycle, for example, a document that is in process will likely be restricted to a small number of authors and perhaps reviewers, while a released document will have a much wider audience. See 2.2.4, "Life-cycles" on page 31 for additional information about life cycles and section 8.4.2, "Security policies" on page 234.

Marking sets

For some data it is desirable to allow the users to modify the security of an object based on its metadata rather than by updating the ACL, for example a document might have a classification level, such as public, secret, or top secret. Each classification conveys a specific set of user access rights. Rather than having to write custom code to update the ACL based on the given property value, the Content Engine offers a *marking set*. Marking Sets can modify the actual rights on the document by removing rights based on the value of a string property.

For more information, see section 8.4.1, "Marking sets" on page 229.

Security Inheritance

Using the IBM FileNet P8 Content Engine's Security Inheritance feature one object, a security parent, can pass its permissions to some other object, its security child. When permissions on the security parent change, the permissions on all its children also dynamically change. This feature gives much easier security management as, for example, you can set up your security so that changing the permissions on a single custom object can affect the security of all documents on your system.

One place where this is used is in folder-inherited security. It works by creating an object-valued property and using this property to propagate security from the target object.

For more information, see section 8.4.4, "Dynamic security inheritance" on page 239.

2.2.3 Event framework

Content Engine provides an extensible framework by which custom code is executed in response to various system- or user-defined events, such as adding a document to an object store.

The primary elements of the Content Engine event framework are:

- Event: A predefined action, such as the creation or deletion of a document.
- Event action: An object associated with the event, which specifies, through its property settings, which custom code to execute in response to the event.
- Event action handler: The code, written as custom Java classes that implement the EventActionHandler interface.
- Subscription: Links one or more events, a target Content Engine object, and an event action object.

Events within the Content Engine can be executed either synchronously or asynchronously to the initiating Content Engine transaction. Synchronous events execute within the transaction context of the executing request, so a failure of the action forces the overall transaction to fail. Asynchronous events are queued for later processing by the Content Engine server in the background, asynchronous-event thread.

System events

The Content Engine allows the firing off of events for certain incidents, which can be subscribed to and handled by a configurable event handler. This ability highlights the event-driven architecture of the IBM FileNet P8 Platform. The Content Engine allows the creation of an event when an important state is changed for a managed object. Table 2-1 lists the system events that can be subscribed to.

Table 2-1 List of system events for the Content Engine

Event	Description	Subscribable class or object
Creation	Triggers when an instance of a class is created or saved or a reservation object is created (CheckOut).	Document, Folder, Custom Object
Deletion	Triggers when an object is deleted from an Object Store.	Document, Folder, Custom Object
Update	Triggers when the properties of an object are changed.	Document, Folder, Custom Object
Update Security	Triggers when the access control information for an object is changed.	Document, Folder, Custom Object
Change State	Triggers when the document life cycle state for a document is changed	Document

Event	Description	Subscribable class or object
Change Class	Triggers when the class of an object is changed	Document, Folder, Custom Object
CheckIn	Triggers when a document is checked in. Only available for Document classes that have versioning enabled	Document
CheckOut	Triggers when a document is checked out. Only available for Document classes that have versioning enabled	Document
Cancel Checkout	Triggers when a checkout is cancelled for a document. Only available for Document classes that have versioning enabled	Document
Classify Complete	Triggers when the classification for a document completes	Document
Promote Version	Triggers when a document is promoted to a new major version. Only available for Document classes that have versioning enabled	Document
Demote Version	Triggers when a document is demoted to a minor version. Only available for Document classes that have versioning enabled	Document
File	Triggers when an object is filed to a folder (including the creation of a subfolder)	Folder
Unfile	Triggers when an object is unfiled to a folder (including the deletion of a subfolder)	Folder
Freeze	Triggers when the Freeze method is called for a document	Document
Lock	Triggers when an object is locked	Document, Folder, Custom Object
Unlock	Triggers when an object is unlocked	Document, Folder, Custom Object

The system events cover a large variety of object-related state changes. Additional *custom events* can be defined, and the subscription to these events and the configuration of the corresponding actions occur in exactly the same way as it occurs for system events.

Custom events

The Content Engine provides various system events that can be used to activate managed content and for auditing purposes. However, in complex situations, the system events that are provided (ready to use immediately) might not provide all of the required functionality. Using the flexible IBM FileNet P8 architecture you can extend the event model by adding *custom events*.

The Content Engine itself does not generate custom events; instead, custom events occur by a custom application through calling a RaiseEvent method for the corresponding object. This approach is beneficial because after the event is raised it is treated like a system event, which means that the corresponding event action (and filter conditions) can be configured using the Enterprise Manager.

Custom event actions

The Content Engine's event-driven architecture can raise system events and custom events. The administrator can configure components that subscribe to these events to take the appropriate action: launching a workflow or by implementing a custom event action.

The event action handler code is written as Java classes that implement the EventActionHandler interface. These custom Java classes are delivered as jar files located through the global class path or stored as content objects called *Code Modules* (the jar files or compiled classes are stored as content elements). When a given action occurs on a particular object, a query is executed to find the set of associated subscriptions and their corresponding event action handlers. For each subscription, the event action handler is loaded through a custom class loader and executed through the EventActionHandler.onEvent() method.

For additional information about creating event actions and developing event handlers, refer to *Developing Applications with the IBM FileNet P8 APIs*, SG24-7743.

Interacting with business processes

Launching a process on the Process Engine is another option for handing an event within the Content Engine. This happens thorough a *workflow subscription*. Workflow subscriptions can be defined and configured using either Workplace XT or the FileNet Enterprise Manager. A workflow subscription calls a defined version of a workflow. It does not automatically launch the latest version for this workflow, which allows transferring and pre-testing a new version of a workflow definition but still maintaining the production workflow subscription using the old workflow definition. If it is required to always launch the latest version of a workflow by a subscription, use a launcher workflow that starts the most current release of the business workflow.

On the level of the workflow subscription, mapping properties for the object that launched the workflow to workflow fields can be configured and additional filter conditions that determine if the workflow will really be launched (for example launch a workflow only if a major version of a document is added).

2.2.4 Life-cycles

Throughout a document's life it moves from one state to another, such as from Application to Approval in the case of a loan application document. The Content Engine provides document life cycle management through life cycle policies and life cycle actions:

- Life cycle polices define the states that a document can transition through. The life cycle policy can define a security template to be applied to the document when it enters a new state. Lifecycle policies can be associated with a document class and applied to subsequent instances of that class or they can be associated with an individual document instance.
- Life cycle actions define the action that occurs when a document moves from one state to another. Life cycle actions are associated with state changes in the life cycle policy object.

In addition to events, a life cycle policy can apply a security policy based on the life cycle stage of a document. For additional information about Security Policies, “Security templates” on page 27.

For additional information about life-cycles, see section 8.4.3, “Document lifecycle policies” on page 237.

2.2.5 Content storage

The primary categories of content storage that are associated with the Content Engine are database storage, file storage, fixed content device storage, and content federation. These storage options can be used individually or in conjunction with one another. A specific storage device is created in the object store as a *storage area*. Individual classes can be configured to use a specific storage area or can be instructed to use a *storage policy*. A storage policy defines the storage area to use based on rules configured for the policy by the administrator.

Content is always streamed from the client to the server and stored in a temporary staging area prior to being committed to the final destination. The final committal step is different depending on the chosen storage option. Figure 2-5 on page 32 illustrates the various Content Engine storage services.

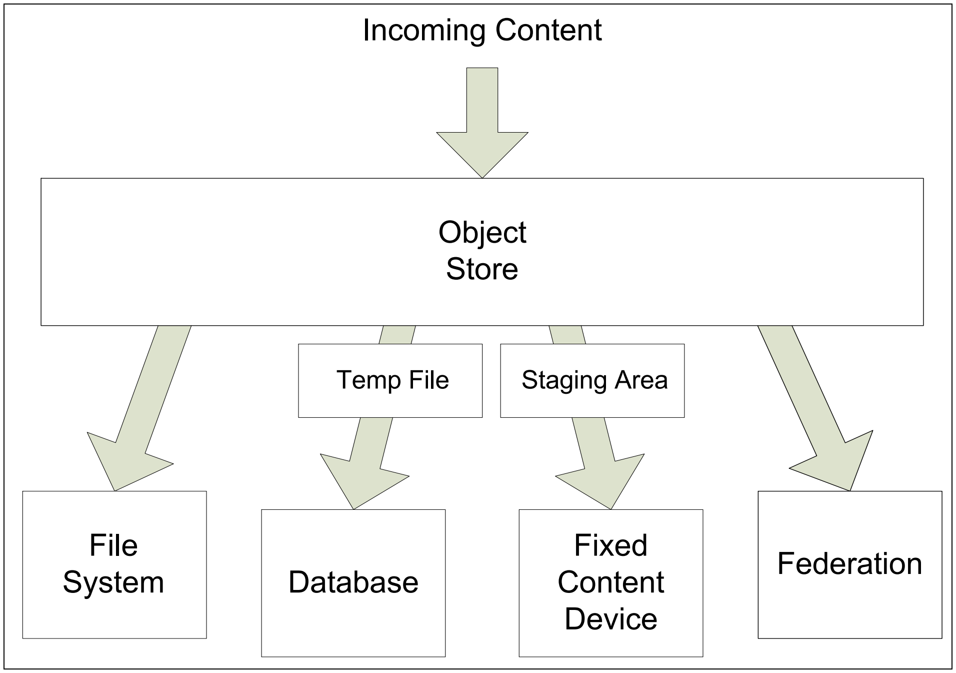


Figure 2-5 Content Engine storage options

Database storage

Database storage is the mechanism for storing content within the configured relational database management system. Each piece of content is stored as a BLOB within the Content table of the database. Database storage is typically configured for content that does not need retention and that tends to be smaller. Every object store has a single database store configured, although it might not be assigned to a class or storage policy.

File storage

File storage represents any device that can be mounted through the Common Internet File System (CIFS) or Network File System (NFS) protocols. The storage device can be direct-attached (either local disk or SAN) or network-attached storage (NAS). All servers in a farmed deployment must have access to all storage areas, which requires that the storage must be shared in some fashion, for example, NAS. Content files are initially written to a temporary file name on the storage device, then the process is completed in the following order:

1. The metadata updates are committed along with an entry in the ContentQueue table.

2. The entry in the content queue table is processed by an asynchronous background thread in which the content file is renamed to its final name.
3. The entry is removed from the ContentQueue table.

Fixed Content Device (FCD) storage

Fixed content devices represent a variety of IBM and third-party products that deliver additional functionality over standard file systems. This functionality can include Write Once Read Many (WORM), hardware-based object retention, multi-tiered storage, and others. An example of fixed content devices supported include the IBM Information Archive, IBM Tivoli Storage Manager, and some third-party storage systems.

Typically, third-party product integrations rely in some part on device APIs to manage the interactions that are necessary for committing or finalizing the content operations. Much like other storage types, the content is streamed from the client and stored in a temporary staging area on the server. The content is then written to a file storage area and a subsequent request to migrate the content out of the queue is processed. This migration request triggers the update to the fixed content device (generally through APIs), after which the system updates the document entry in the DocVersion table with a referral record to the entry in the FCD, and deletes the file in the staging area.

Content Federation

The final storage medium for a Content Engine system is federated document storage. Federation allows for the Content Engine to access data stored in a third party or mature system, such as IBM FileNet Image Services or third-party systems through IBM Content Integrator. For more information about federation, see Chapter 5, “Expansion products for connection and federation” on page 117.

Storage policies

In addition to defining where content can be stored in a repository, the Content Engine also offers an additional level of abstraction and definition to content storage through *storage policies*. A storage policy defines target storage using rules and can be used to farm content out to a number of storage areas for load balancing.

Using a storage policy allows for additional storage areas to be added to a system at a later date and ensures that all new content (new document and new versions of existing documents) uses that storage. Each class that accepts content can be defined with either a default storage area or a storage policy. If a storage policy is defined, at object creation time the system determines the final storage area based on the rules of the storage policy.

2.2.6 Full-text indexing

Content Engine provides search services based on the metadata of an object. Textual data and property values pertaining to documents are searchable using Content Search Services or the Legacy Content Search Engine.

Content Search Services

With the P8 5.0 release, Content Search Services (CSS) is introduced as an alternative to the IBM Legacy Content Search Engine (CSE). CSS uses Apache Lucene for indexing and search, UIMA (Unstructured Information Management Architecture) for tokenization and, LanguageWare® for dictionary processing. Full-text indexing and indexing of specific metadata attributes can be configured on a per class basis, for instance, any create or update operation on a given document instance triggers the server to evaluate the object's full-text attribute to determine whether to queue the document for indexing.

Each CSS instance can be used for both indexing and searching tasks. You can deploy a single CSS server in your environment to be used for both indexing and search. However, the recommended approach is to deploy multiple CSS server instances and assign each one to do only indexing or only searching tasks. See Figure 2-6 on page 35 for these deployment options.

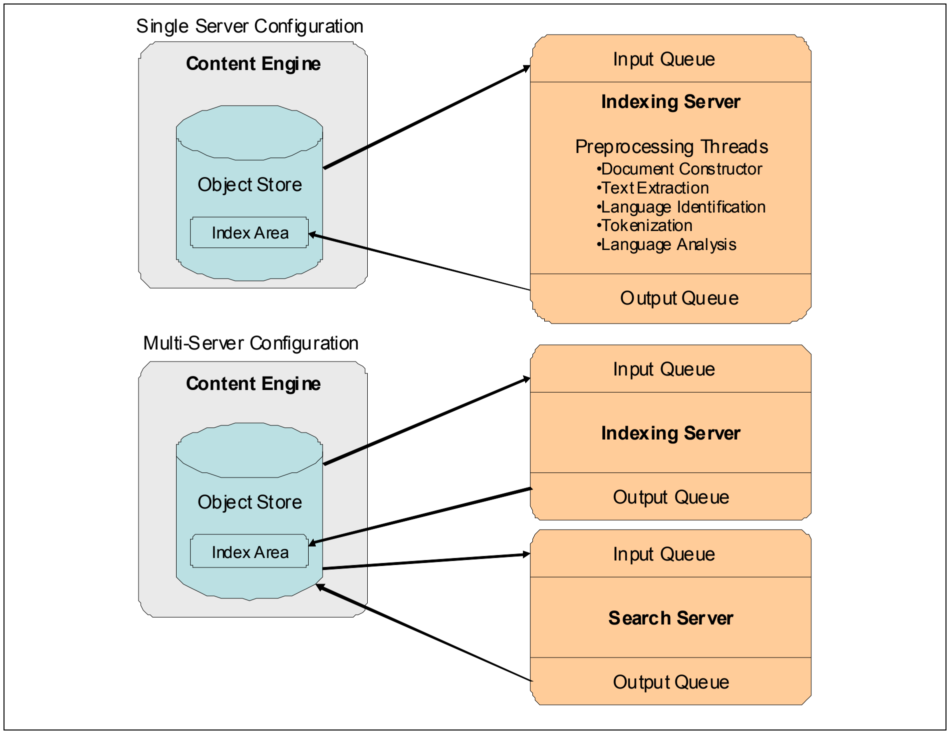


Figure 2-6 Content Search Services server configurations

The indexing process begins at the Content Engine when CBR-enabled objects, such as documents whose class is CBR enabled, are created or updated. The Content Engine stores indexing data for the object in indexes managed by the CSS servers. Each index is associated to a distinct index area in an object store. During the indexing process, the system can write to multiple indexes across these different index areas. When an index's capacity is reached, the index is automatically closed and a new index is opened. The Content Engine runs a background service on an indexing table to identify documents that are queued for indexing. The system queries items from the indexing table and then groups index requests pertaining to the same target index into an index batch. The binary documents in this batch are converted to text, and the entire batch is then submitted to a CSS indexing server.

After the index server receives the index batch, the preprocessing functions begin. Preprocessing functions consist of document construction, language identification, and tokenization of the document's content and properties:

1. The XML filtering utility removes any superfluous XML elements from the extracted text containing XML elements. Non-searchable XML elements must

be defined as a surplus element in the CSS configuration. This feature assists with minimizing index files by not indexing XML elements that will not be relevant for full text searches.

2. The language of the text is identified. To ensure accurate processing and optimal performance, a default language can be specified for an object store. If one language cannot be definitively identified for the content of an object store, the identification of the language can be set to automatic and triggers language identification on an object-by-object basis.
3. In the final step of preprocessing, tokens are created from the extracted text and index for the document is updated or created in its respective index area.

Users can submit Content-based retrieval (CBR) queries against the metadata, the full-text index data, or both. Search requests are initiated through FileNet Enterprise Manager or other client applications using the Content Engine API and includes a full-text expression that is submitted to the search server by the subsystem dispatcher. The search server can use word stems, synonyms, and stop words to improve search efficiency. The search server searches for and identifies the stem for all word terms included in a full-text search expression, for example, if you search for the word “running”, the search server automatically searches for the word “run” too. The search server also searches for all defined synonyms in the search expression, for example, if “International Business Machines” and “ibm” are defined as synonyms, searching for “ibm” is equivalent to searching for “ibm” or “International Business Machines”. A stop word is a word or phrase that is ignored by the search server to avoid irrelevant search results caused by common expressions, for example, if the word “the” is defined as a stop word, the phrase “The cat and the dog” and “cat and dog” are equivalent. The search server uses these definitions to evaluate the appropriate index and runs the full-text search. The results are returned to the subsystem dispatcher which then joins the results with other tables in the query and runs the query.

Legacy Content Search Engine

The Legacy Content Search Engine uses the Verity K2 server, a search engine from Autonomy, to provide full-text indexing. The Content Engine stores indexes into collections, where each collections is managed by a Verity server instance. During the indexing process, the system can write to multiple collections depending on the indexes being create or updated. Each index area has its own distinct set of collections. When the collection’s capacity is reached, the collection is automatically closed and a new collection is opened. The Content Engine runs a background service on an indexing table to identify documents that are queued for indexing. The system queries items from the indexing table and submits them in a batch to the Verity server. The Verity server uses a location on the file system to temporarily store documents before they are indexed. If a given document has any metadata attributes tagged for indexing, those attributes are written to a separate text file that is passed to the Verity server along with the content files to be indexed. Metadata and content files are passed to the Verity server as URIs, which the search engine uses to directly access and read those files for indexing.

Users submit CBR queries to CSE from the FileNet Enterprise Manager or custom application using the Content Engine API. Results from queries against the metadata and full-text index data are dumped into a temporary table and a join is executed across the associated metadata table to handle any metadata-related portions of the query. Where multiple, active, index areas are configured for a given class of objects, queries are executed against each index area, and the final result-set is aggregated from the individual query and returned to the Content Engine. CSE also uses the concepts of word stems, synonyms, and stop words, as described in the Content Search Services section, to improve search efficiency.

Migration

The Content Engine allows for the migration of Legacy Content Search Engine (CSE) indexes to a Content Search Services object store. Migration is initiated by enabling CSS in an object store that is already enabled for CSE. A background migration index job is launched and processes all objects indexed in Verity collections and indexes them into CSS indexes. The Content Engine runs in dual-index mode, where objects are indexed, updated, and deleted in both Verity collections and CSS indexes. Dual-indexing is required until all CSE objects for the respective object store are migrated to CSS, but the Content Engine can run in this mode for as long as the administrator deems necessary. Searches are routed to CSE, but there is an option to route searches to CSS on a per-search basis to allow for testing of the CSS indexes (CSS indexes are incomplete until the migration process is complete). After completion of the migration index job, the default search engine can be switched to CSS. Switching to CSS affects all client application searches and can be reversed prior to disabling CSE from the respective object store. After the switch to CSS is successfully validated, CSE can be disabled, which ends dual-indexing mode. Disabling CSE is a non-reversible action.

2.2.7 Publishing

Content Engine provides publishing services using integration with a third-party rendition engine from Liquent. The rendition engine provides the ability to render various documents’ formats into either PDF or HTML. The publishing framework can be leveraged to generate a new publication document or to generate a new version of an existing publication document.

The publishing framework consists of two primary components, a publish template and a publish style template. Along with the IBM FileNet Enterprise Manager, the Publishing Style Template Manager is installed. This tool can be used to create and edit publish style templates. The Workplace-supplied Publishing Designer is used to create and edit publish templates, which optionally can specify a publish template. The publish template is used during the publishing operation and defines the properties and security attributes for the published document. The publish style template defines the output format and various other rendering options, such PDF security and PDF watermarks.

2.2.8 Classification

The Content Engine provides an extensible framework that enables incoming documents of specified content types to be automatically assigned to a target document class and setting selected properties of that target class based on values that are found in the incoming document.

When a new document is created, a flag determines whether automatic classification is executed or not. If classification is enabled for the document, the Content Engine executes the following steps:

1. The classification is performed asynchronously by queuing a classification request (with the reference to the document). The Content Engine ensures that the classification requests are properly queued and that the status for the classification is set to “pending” for the document.
2. Within a transaction:
   - The Document is handed to the Classification Manager, which determines the MimeType from the source document.
   - The Classification Manager determines which Classifier must be invoked for the MimeType and passes control to the Classifier.
   - The Classifier extracts the information from the document, performs a Change Class operation based on extracted information, updates the metadata accordingly, and passes back a status.
   - The Classification Manager evaluates the status, sets the document’s classification status accordingly, and deletes the request from the queue. The Content Engine ships with one default auto classification module for XML documents.

In addition to the internal classification system, there are a number of add-on products that can assist in performing content classification. For more information, see Chapter 6, “Expansion products for Information Lifecycle Governance” on page 135.

2.2.9 Protocols

A Content Engine deployment might take advantage of several protocols. On the client-side, requests come into the client tier through the web service listener or the EJB listener. Although the web services listener communicates through HTTP only, the EJB listener communicates using the same protocol that the application server leverages for its EJB interactions (IIOP, for example). Workplace XT communicates to back-end Content Engine servers through the EJB protocol only. Custom clients can use either the EJB or WSI interfaces. IBM FileNet Enterprise Manager (the Content Engine administration client) uses the .NET API over the web services (HTTP) interface.

On the server-side, the Content Engine uses:

- JDBC to communicate with the relational database management system, including the database storage areas.
- NFS or CIFS to communicate with the file storage areas.
- Various protocols for the fixed content devices. For more information, see the third-party vendor-specific documentation.
- LDAP to communicate with the underlying directory service provider(s).
- Liquent rendition engine's proprietary protocol.
- Content Search protocol

2.2.10 APIs

The Content Engine offers two primary APIs: Java and .NET. The two APIs are almost identical and offer objects and methods to handle creating, retrieving, updating, and deleting objects, folders, and documents. In addition, the API offers methods for handling additional administrative tasks, such as updating security.

While the two APIs are almost identical, the Java API does offer the additional ability of being enlisted in a J2EE transaction and the ability to communicate over the native application server protocol, such as IIOP. However, the Java API can communicate over the web services transport just like the .NET API.

For additional information about developing with the Content Engine API, refer to *Developing Applications with the IBM FileNet P8 APIs*, SG24-7743.

2.2.11 CMIS

IBM FileNet P8 5.0 now offers support for the Content Management Interoperability Services (CMIS) standard. This standard includes both an object model and two bindings (a RESTful and a web service based binding). While the CMIS standard does not expose the complete capabilities of the Content Engine, it does provide a standard set of functionality across all compliant repositories. The IBM CMIS server's available functionality from the specification includes:

- Support for all CRUD (Create, Read, Update, Delete) operations on documents and folders
- Versioning (checkin, checkout)
- Filing operations for documents in folders
- Type (metadata discovery)
- A powerful SQL-92 based query language

2.3 Process Engine

The Process Engine provides the IBM FileNet P8 Platform with workflow management capabilities. These capabilities include personal inboxes, work object routing, group queues, process tracking, and process orchestration capabilities.

As of version 5.0, Process Engine is now a pure Java-based application that stores all workflow information (except the raw process maps that are stored in the Content Engine) in a database. To help facilitate logical separation between lines of business and applications, the Process Engine offers the ability to segregate workflow data into subdivisions of the repository called isolated regions.

2.3.1 Architecture

With the move to a Java-based implementation, the Process Engine was migrated to a single process, multi-threaded architecture. This move allows for better use of the operating system, its resources, and simplifying start up and shutdown routines.

Another benefit of the move to a single process and away from the shared memory model of the old implementation is the possibility of multi-tenancy within a Process Engine server. Now it is possible to have multiple instances of Process Engine fronting different process stores and running on the same system. There is no support for multiple server instances on the same system fronting the same Process Store. Also there is no support for a single Process Engine server instance that fronts multiple Process Stores.

The Process Engine has two kinds of processes: PEServer and PEManager. A running Process Engine Server instance is an instance of the PEServer process. It handles all database persistence, background tasks, daemon threads, and fields any RPC calls. The PEManager process runs as a single instance in a system. It is responsible for managing the Process Engine server instances including starting and stopping them.

A particular Process Engine is configured as a member of a given Content Engine domain and provides workflow management for any applications and users that need access to the data in that domain. A given Process Engine service can be leveraged by one or more Content Engine object stores for managing business processes that are associated with content in those repositories.

Figure 2-7 shows the Process Engine system architecture.

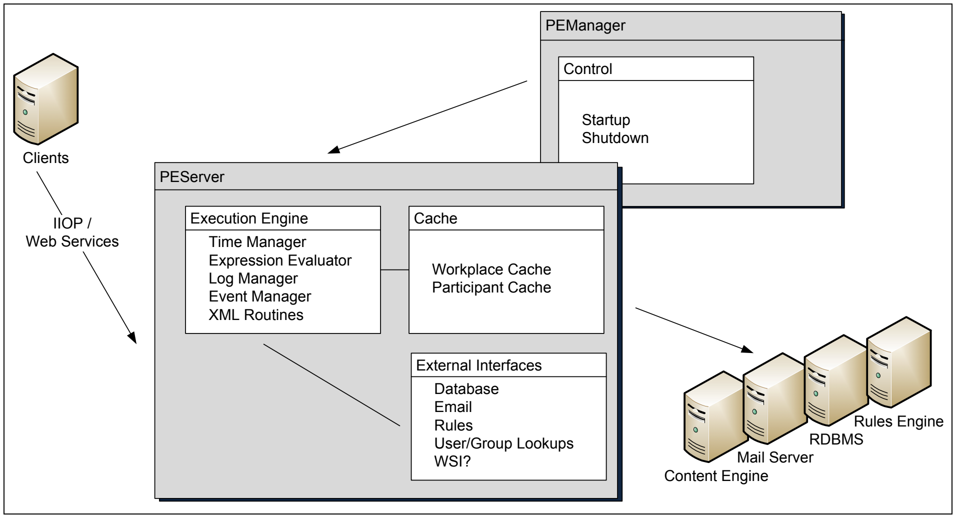


Figure 2-7 The Process Engine system architecture

2.3.2 Data model

Process Engine divides workflow data into work items in rosters and queues. Workflow data and transferred workflows are stored in a RDBMS. This database is either stand-alone or co-located with the Content Engine database. For Case

Manager, the database must be co-located with the Content Engine database; otherwise, the database can be separate.

Isolated regions

The Process Engine allows process data to be segregated into smaller units within a single process store. This division is called an *isolated region*. An isolated region contains all of the related process definitions and metadata in separate tables within the Process Engine database. It also contains a unique set of rosters, queues, and work items.

Workflow definitions

Workflow definitions are made up of maps. Each map contains steps and routing logic. They are stored in the Content Engine, and are then transferred and compiled in the Process Engine. Built using Process Designer, described in section 2.5.2, “Process Engine” on page 54, the workflow map defines the steps, routing, and participants in a workflow.

Work items

When a workflow is launched, a new *work item* is created. A workflow instance can contain one or more work items. The work item does not contain any map or definition information. It has information, such as the current map and step ID. The work item contains the properties for the work item. These properties can be of the following types: Integer, String, Boolean, Float, Time, Attachment, and Participant.

An integral part of the handling process is often a piece or pieces of content. This content must be carried with the workflow so that the user or system acting on that step can view or retrieve the content in question. The work item contains a reference to one or more content objects through the use of attachments. In the context of a IBM FileNet P8 deployment, these attachments are references to objects within the Content Engine.

The metadata and attachment data is serialized and stored as a BLOB within the database. While all data is stored in the BLOB, Process Engine does offer a way to expose some of the data in a searchable format. See “Rosters” and “Queues” on page 43 for more information.

Rosters

After a workflow is instantiated, the data is available in a *roster*. The process map defines which roster a given process definition is associated with at design time.

The roster exposes a number of system-level fields, such as date created, and can expose selected properties from the work item to help speed finding work items. At runtime, when a work item is created or updated, the values in the exposed properties are copied into the database table.

A given Process Engine isolated region has at least one roster, the default roster. In addition, more rosters can be created to help segregate data and to improve performance. Each roster that is created results in a new table being created in the database and all work items assigned to that roster referenced in that table.

Rosters are used to query for objects regardless of where they are in their process flow, typically for administrative purposes.

Queues

While work items can be retrieved from the roster, it is important to offer a way for users or applications to find work items that are relevant to them. Process Engine offers an additional level of organization in the form of a process queue. A *queue* represents all of the running instances that are on a particular step (or related steps) within the work item. Where the roster represents objects that are based on their type of process, a queue represents objects that are based on the type of step they are on, at a particular point in time.

There are different types of queues:

- Process queues can be thought of as public queues where numerous users might have access and any of those users are allowed to browse and process work items from those queues on a first-come-first-served basis.
- User queues, in contrast, are associated with a particular user and only have work items that are specifically designated for that user to work.
- Component queues are a special kind of process queue that the Component Manager application uses for background processing of custom actions.
- System queues manage various system activities on work items.

In-baskets

In-baskets provide an additional way to display items from a queue, basically a view to a queue. This view can have one or more filters to apply business rule-based criteria to the items in the queue. These filters cannot be modified by the user, only by the process administrators. In addition to filters, in-baskets allow for the definition of columns or sets of attributes from the underlying queue. These columns are also made available to the user or application.

Roles

A role is used within Process Engine to tie a set of users and groups to one or more in-baskets. This provides an application developer or an application, such as widgets, a simple way to present users the in-baskets that are appropriate to their position with an organization.

Application spaces

Application spaces within the Process Engine provide a solution with a way to group together a number of roles that apply to that application or line of business.

Event Logs

Event logs are used to log event data for various activities that occur on work items within the Process Engine. Like rosters, there is a default event log and additional event logs can be configured. Process definitions can be associated with a given event log so that subsequent actions that occur on instances of those process definitions can be recorded. Event logs, like rosters, are separate tables within the Process Engine database. Each row represents an event action for a specific work item. Also like rosters (and queues), the metadata that is collected in these tables is defined by the columns that are configured for a given event log. The column name and type are matched with corresponding properties of a work item. For those that match, the data is copied from the work item.

The types of events that get logged are configurable and provide a fair degree of control over how much data is collected and what it represents. The event log table can be queried to retrieve data on historical events that occurred within the system. The Process Tracker application uses this data to show the history for a given work item. The event log tables are also the primary source of data for the analytics engine.

Figure 2-8 on page 45 shows some of the tables in the Process Engine database.

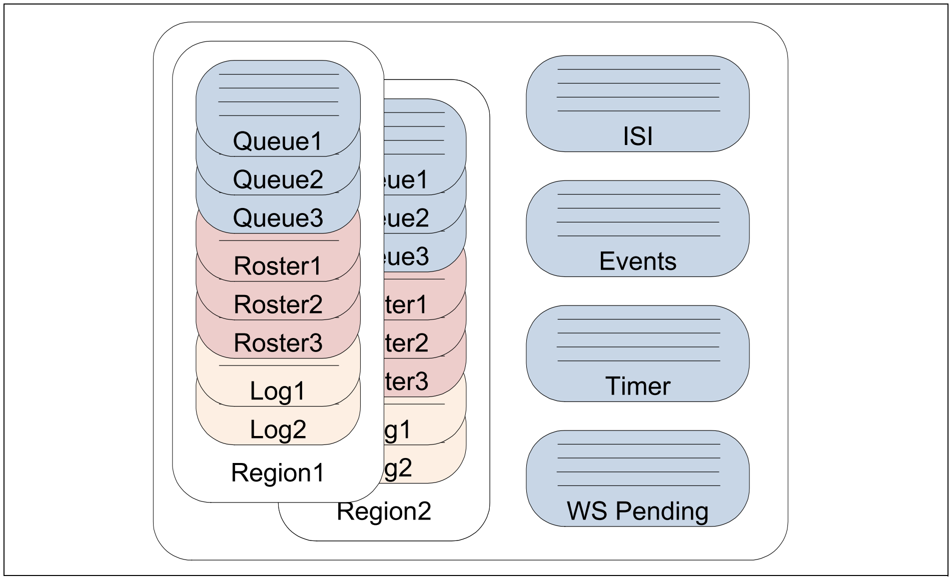


Figure 2-8 Process Engine database

Accessing work

Any given work item can be retrieved and examined through a variety of views. These views are roster element, queue element, step element, and work object:

- A *roster element* represents each of the data elements (columns) exposed for a given roster.
- A *queue element* represents each of the data elements (columns) that are exposed for a given queue.
- A *step element* represents each of the data elements defined for a given step in the process flow, which are defined at design time by specifying for any given step what fields are used. Step element data fields might be dynamic in the sense that the definition for a step might include an expression for the data element that is evaluated at runtime when generating the step element.
- A *work object* represents the data defined in the workflow definition. Each work object is a copy of the data. If a workflow instance contains multiple active work objects, each work object might contain different data.

Understanding this model is important for performance and scalability reasons. Roster and queue elements are generally the fastest and lightest-weight elements because they represent a subset of the overall metadata, and neither requires deserialization of the binary data to generate. Step elements usually offer the next best performance. They do require deserialization of the binary data but also often represent a small subset of the overall metadata. Work objects are the most expensive because they require deserialization of the binary data, and return all metadata associated with a given work item.

2.3.3 Access control

Security within the process system is not managed on individual work items but rather on the queues and rosters in which these items reside. User access to work items or portions of work items is controlled by the security rights that they have on the associated rosters and queues that those work items exist in. User queues are additionally enforced by giving access only to the user for which those work items were specifically assigned (based on the current step in the process flow). For more information about Process Engine security, see 8.2.2, “Process Engine” on page 210.

2.3.4 Process orchestration

The Process Engine can be either a provider or a consumer of web services. One mechanism for accomplishing this is through the process orchestration framework. Based on the orchestration portion of the BPEL specification, this framework, shown in Figure 2-9, provides a mechanism by which individual process steps can call out to an external web service or, conversely, be exposed as a web service for external consumption.

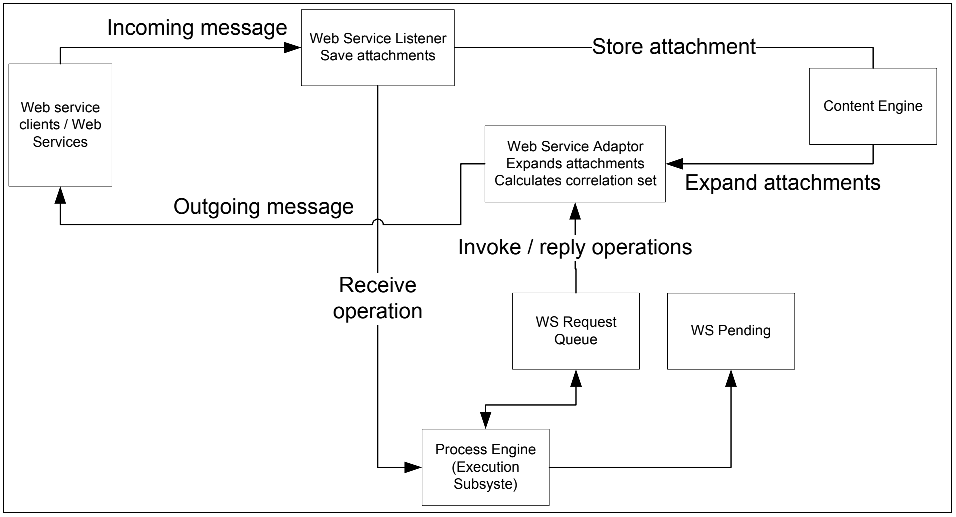


Figure 2-9 Process orchestration

There are three main actions involved: *receive*, *reply*, and *invoke*. The receive and reply steps define a point in the process to expose externally as a web services entry point and if necessary return a response. The invoke and receive steps call out to an external web service and if necessary receive a subsequent response. Figure 2-10 depicts the various interactions that are enabled through the process orchestration framework.

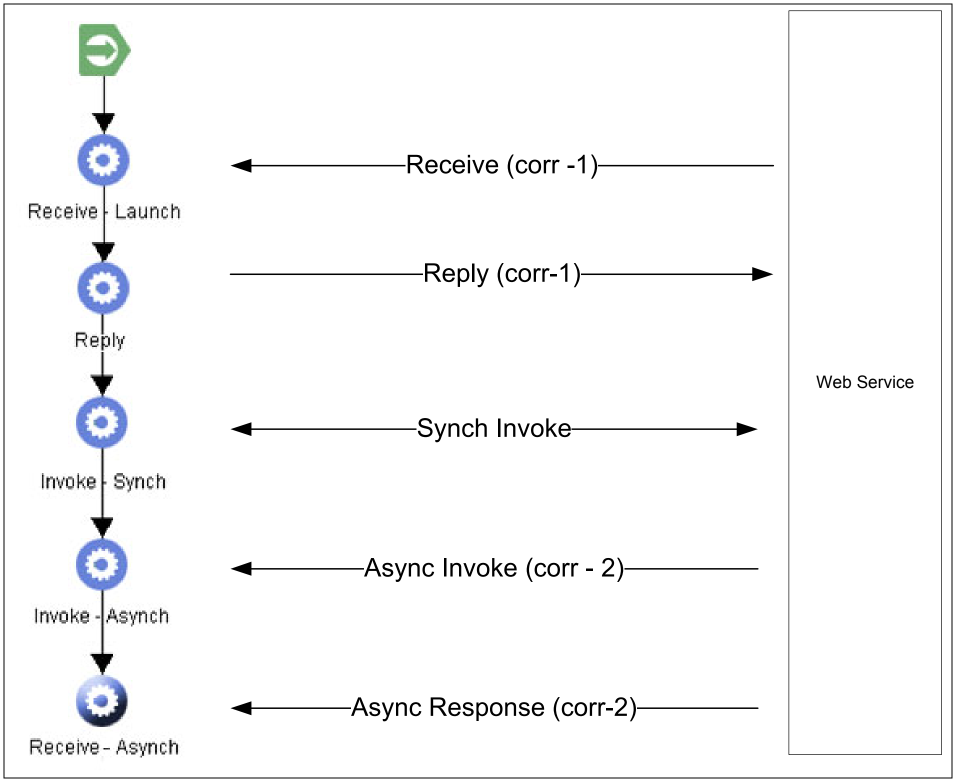


Figure 2-10 Process orchestration interaction

2.3.5 Component Integrator

The Process Engine also offers a service for integrating with external systems. The *Component Integrator* makes it possible to import custom Java components and make them available in a workflow. In the workflow definition, a component step connects to a component queue configured for one or more operations in the external component.

There are two parts to the Component Integrator:

- Component Manager: This service runs on the Workplace/Workplace XT and connects a work item with its appropriate Java or JMS Service Adaptor.
- Configuration: Configure the component queues using Process Configuration Console.

2.3.6 Analysis and optimization

As opposed to business activity monitoring, business analysis and optimization focuses on data that is gathered over a time in the past. The time span being analyzed might be days, weeks, or even years to analyze trends and draw conclusions.

IBM FileNet P8 Platform supplies the Case Analyzer tool to analyze business processes. Case Analyzer leverages Microsoft MS SQL Analysis Services to supply the data in a format that can be quickly explored and drilled down by users. The Process Simulator tool is also available to perform what-if simulations of the process model to discover bottlenecks in the process execution. Analysis and simulation aim for continuous improvement in the quality of the business processes. For this optimization, it is necessary that the process is executed on a BPM system to efficiently collect the matrix.

Data from the Process Engine event logs is fed on a schedule into a datamart database. This datamart stores the data in a special representation (snowflake/star schema) as opposed to the flat schema that is used by the event logs' tables, for example. In a second step, the OLAP cubes are calculated from the current datamart information. There are basic OLAP cubes, which are provided during the installation of the Process Engine, and customers can define new cubes, if required. The configuration of the cubes is stored in the Microsoft SQL Analysis server. Figure 2-11 on page 49 outlines how data from the Process Engine event logs becomes available in the OLAP data cubes for further investigation.

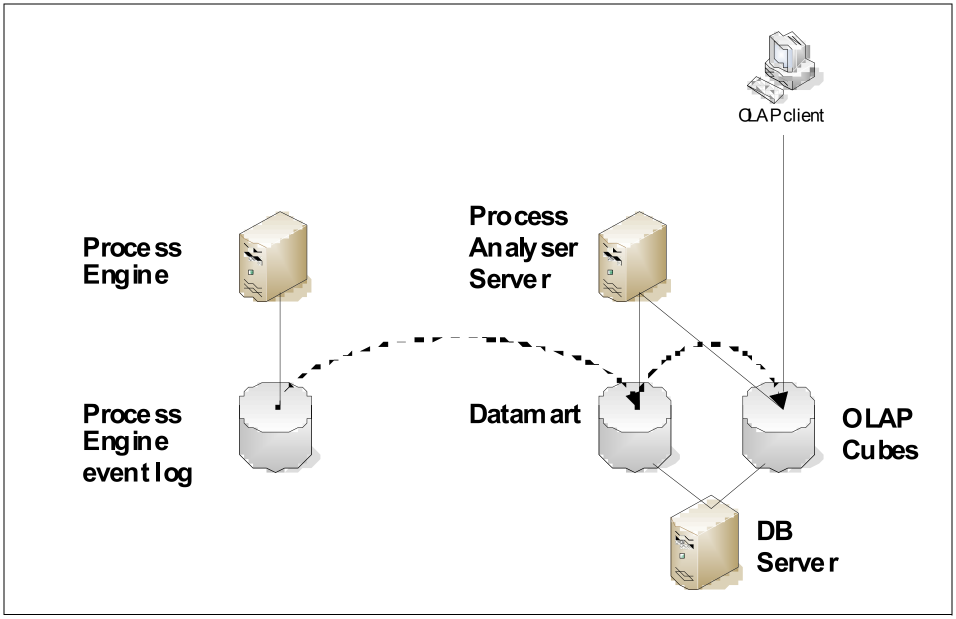


Figure 2-11 FileNet Process Analyzer data flow

The OLAP cubes can be inspected using an OLAP client, for example, IBM Cognos Business Intelligence or Microsoft Excel. Case Analyzer installs a number of predefined Microsoft Excel spreadsheets that use the base OLAP cubes to generate reports for information, such as process execution time, step completion time, queue load, and much more. Use the reports to perform a *slice and dice* analysis, which means that the data viewed is narrowed down further to see details, for example, a report might show the number of completed transactions over a time period, and then how the transaction value affected the processing time or how the transactions are distributed over different regions and if there are differences in the average transaction value for the regions. Assuming that these details are gathered from the event logs and are stored in the datamart, these analysis scenarios can be performed by just a few clicks.

Based on the results of the analysis, the Process Simulator (PS) can be used to determine which changes to the process definition must be applied to eliminate given inefficiencies or bottlenecks. To do so, the process definitions are reused, altered, and loaded it into the PS. For each simulation, a scenario is defined that consists of the process model, arrival times, work shifts for manual processing steps, and (optionally) costs. Arrival times can be defined manually or derived from the production Process Engine. The scenario is versioned in the Content Engine and handed over to the PS. The PS uses statistical methods for the arrival times and simulates the flow of the process instances on the workflow map. The PS provides basic measures for the simulated process, such as process execution times and costs. In case a deeper analysis of the simulation is desired, a Case Analyzer instance can be attached to the PS. In this configuration, it is possible to further analyze the simulation results with the Case Analyzer, as previously described.

The optimization of Content-centric processes is an important building block in the architecture of an ECM platform. Continuous optimization of the business processes is a critical asset for organizations to keep their competitive advantage even in changing business conditions. Leveraging the flexibility and the support of a wide range of standards of IBM FileNet P8 Platform allows enterprises to roll out solutions for their Content-centric processes that can easily be adjusted to changes in the way they perform their business, thus ensuring a low TCO compared to applications that are individually developed.

2.3.7 Rules framework

Rules Engine leverages industry standard web service as the communication mechanism for external application to invoke their business rules. With the Process Designer, the customer can author web service calls to the Rules Engine from the workflow. At run-time the workflow invokes the business rules as part of the process.

2.3.8 Protocols

There are a variety of protocols leveraged within a Process Engine deployment. Starting at the client tier, requests might come in through various out-of-the-box applications that make SOAP requests over HTTP to the Workplace or Workplace XT server. Custom applications that take advantage of the Java API communicate over IIOP to the Process Engine server. Lastly, there are a set of exposed web services that can be accessed through HTTP.

After it is in the server, the Process Engine uses the protocols:

- JDBC connectivity to the database
- Web services calls over HTTP to Content Engine to resolve users and groups

2.3.9 APIs

There are three primary APIs for the Process Engine, the Java API, the REST API, and the web services API. The Java API communicates with the Process Engine over IIOP.

The APIs provide classes and methods for the following functions:

- Creating, working, and completing workflow items
- Searching for work items in queues and rosters
- Viewing the event logs

The REST and web services API uses the Java API to communicate with the Process Engine.

For more information about the API, refer to *Developing Applications with the IBM FileNet P8 APIs*, SG24-7743.

2.4 Workplace and Workplace XT

Workplace and Workplace XT provide user interfaces that can immediately access Content Engine and Process Engine. These web applications are also a container for a number of additional add-on elements necessary for operation of an IBM FileNet P8 system.

Workplace is the official name for the Workplace web application, which is different than the Workplace XT web application. They both offer similar functionalities but have their differences. The preferred user interface is Workplace XT.

Workplace XT is a JSF-based application that provides a general folder-based view of an IBM FileNet P8 content repository. In addition to the content management functions, Workplace XT provides user interface for process management items, such as inboxes, public queues, and step processors. Finally, Workplace XT hosts a series of Java applets and wizards for basic application deployment, search definition, and UI component definition. For more information about Workplace and Workplace XT, see 3.1, “Workplace XT” on page 64 and 3.2, “Workplace” on page 67.

> **Note:** Workplace can also host custom applications that use the Web Application Toolkit provided by Workplace only or the Content and Process APIs that are readily available.

Application Engine and Workplace naming convention: Application Engine is the official name for Workplace. Application Engine does not equate to Workplace XT. Both Workplace and Workplace XT support a common set of functions but differ in other areas. For consistency of the terminology used in the book, we use Workplace instead of Application Engine throughout the book.

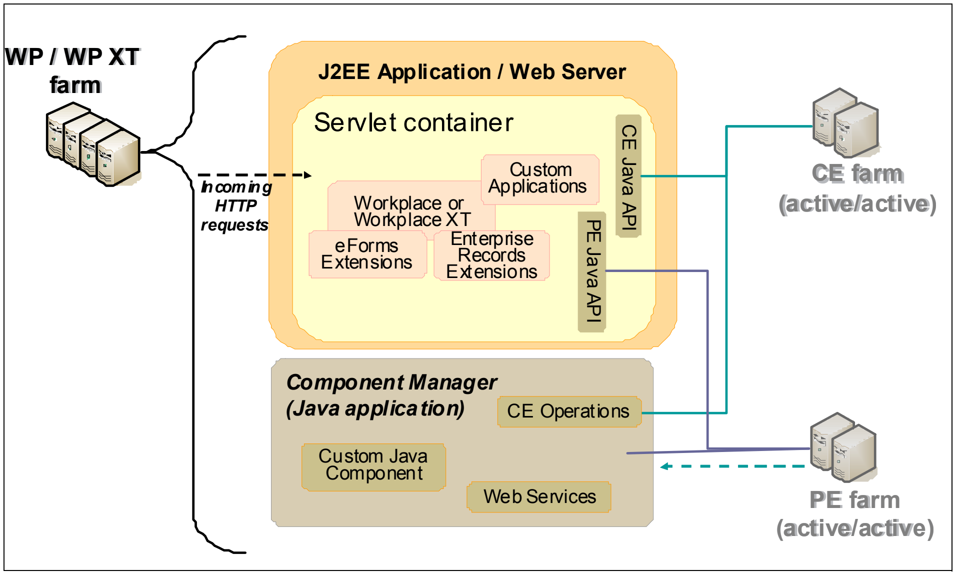


Figure 2-12 Workplace/Workplace XT system architecture

2.4.1 Component Manager

The Component Manager is a Process Engine component that is hosted and managed within the Workplace/Workplace XT. The Component Manager provides an integration framework that enables the Process Engine to make calls on external components. These might be Java components, web services, or JMS queues, for example, the Component Manager includes a web services adaptor that handles outbound web services calls for the process orchestration framework. It also handles calls on custom Java components that can be registered within the system and called at explicit points in a process. The Component Manager also hosts the CEOperations component, which can be used for executing certain content-related operations against the Content Engine.

2.4.2 User preferences

User preferences are a collection of configuration settings that allow a user to modify various display and functional characteristics of Workplace or Workplace XT. Those preferences are stored within a user preference object in the Content Engine repository and retrieved at logon to customize the user experience.

2.5 Configuration tools

For each of the primary engines within an IBM FileNet P8 deployment, there are a number of configuration and management applications.

2.5.1 Content Engine

The Content Engine offers three primary configuration and management interfaces: Enterprise Manager, Administrative Console for Content Engine, and Configuration Manager.

IBM FileNet Enterprise Manager

IBM FileNet Enterprise Manager is the configuration and administration tool for Content Engine. IBM FileNet Enterprise Manager is a Microsoft Windows® application built using the .NET API and communicates with the Content Engine using the web services interface. IBM FileNet Enterprise Manager supports the following actions:

- Configuring all aspects of the domain and underlying object stores.
- Defining custom metadata, such as classes, properties, templates, subscriptions, and event actions.
- Assigning many aspects of security access rights.
- Searching for and administering instances of documents, folders, and custom objects.

IBM Administrative Console for Content Engine

Finally, in IBM FileNet P8 5.0 there is a new web-based domain administration system, the IBM Administrative Console for Content Engine (ACCE). The ACCE offers a subset of the domain administrative tools, including:

- Initial domain configuration
- Object store creation and security management
- Work with PE connection points

- Trace control and logging management
- Review of Content Engine system health metrics

IBM Configuration Manager

The IBM Configuration Manager is designed to help with common setup and configuration tasks. These tasks include configuring the underlying application server JDBC data sources and providers, creating a new Content Engine domain, and creating of new object stores.

2.5.2 Process Engine

There are many tools that the Process Engine provides for designing processes, administering process instances, configuring the data model, administering the server functions, and doing low level analysis of various server activities.

- Process Designer provides the general process design capabilities where users (typical business and IT analysts) define their process flows. The Process Designer can read and write proprietary format (PEP) and XPDL 2.x workflow maps. In addition, Process Designer can import Visio documents, mapping shapes to native objects. A provided Visio template provides BPMN shapes.
- Process Administrator lets an administrative user query the system for process instances and view the current state of those instances.
- Process Tracker can be launched to view the current and historical state of an individual process instance.
- Process Configuration Console defines the rosters, queues, event logs, and various other system-related components.
- Process Task Manager can be used to start and stop the various server components, including the server itself.
- There are a series of lower-level tools on the server, such as vwtool and vwlog, that can be used for viewing detailed information about server state and activities that occur there.

2.5.3 P8 Platform tools

The IBM FileNet P8 Platform also includes a couple of cross-engine tools for configuration and deployment. These tools include the IBM FileNet Deployment Manager and IBM FileNet System Dashboard.

IBM FileNet Deployment Manager

The IBM FileNet Deployment Manager is a tool that moves content and processes solutions from one environment to another. In a typical environment, all development and testing occur on different environments than the production. This setup allows for sandboxing development and ensuring changes are properly tested before bringing them live in production.

To help simplify the movement of classes, properties, and workflows between, the Deployment Manager takes a set of objects from the source system, compares them to the target system, and either creates or updates the target system with the changes.

IBM System Dashboard for Enterprise Content Management

The IBM System Dashboard for Enterprise Content Management provides a number of system management and performance reporting tools for the various engines in the IBM FileNet P8 Platform. The System Dashboard can gather and display information, such as:

- Operating system information and performance data
- IBM FileNet-specific data, such as RPC counts, Content Engine operations, and other engine specific counters
- Environmental data including Java runtime versions and memory settings

In addition to gathering the counter and system information, the System Dashboard provides the ability to create user-defined charts, export data for historical analysis, display alerts and urgent messages from applications, and provides listener agents for gathering data from other applications.

2.6 Case Manager

IBM Case Manager, new for IBM FileNet P8 5.0, is a platform for case management that is built into the IBM FileNet P8 Platform. Case Manager is a more flexible solution for solving business problems. Through a combination of a widget- and Web-based UI, a flexible task-based environment, and comprehensive case-based data model, Case Manager enables users to more flexibly complete their business goals.

Case Manager offers a different approach to solving problems by enabling a wider range of groups within an organization to assist in creating the solution. Instead of just relying on developers to build the solution, Case Manager functionality and user interfaces allow business analysts a greater role in helping define the types of problems being solved and how they are solved within an IBM FileNet P8 solution. With that in mind, Case Manager and is designed to be:

- A platform for designing and building case solutions
- A run time environment for launching, processing, and interacting with cases
- A set of tools for configuring and moving solutions into production environments
- A set of APIs and templates for customizing case solutions

In addition to being built on the core platform, Case Manager also offers a native case object model built into P8 Platform. This object model is extensible and allows for defining the various elements of a case and how they are to be handled. See 2.6.1, “Data model” on page 57 for more details about the Case Manager data model.

To help facilitate rapid deployment of solutions, Case Manager comes with a widget and web-based user interface for both design and runtime use. Case Manager also offers a REST-based API for building custom solutions that go beyond the functionality offered by the user interfaces that are readily available.

Case Manager also offers deep integration with the following components:

- WebSphere Integration Developer 7 Feature Pack 2 which enables integrating WebSphere Process Server components and processes with case management tasks.
- Content Analytics 2.2 (Full Text Unstructured Analytics)
- Cognos Real Time Monitoring 10.1 (Active Analytics)
- Cognos BI 10.1 (Historical Analytics)
- Lotus Sametime® 8.5.1 (embedded awareness in case runtime and web chat)

Figure 2-13 on page 57 shows the architecture of Case Manager.

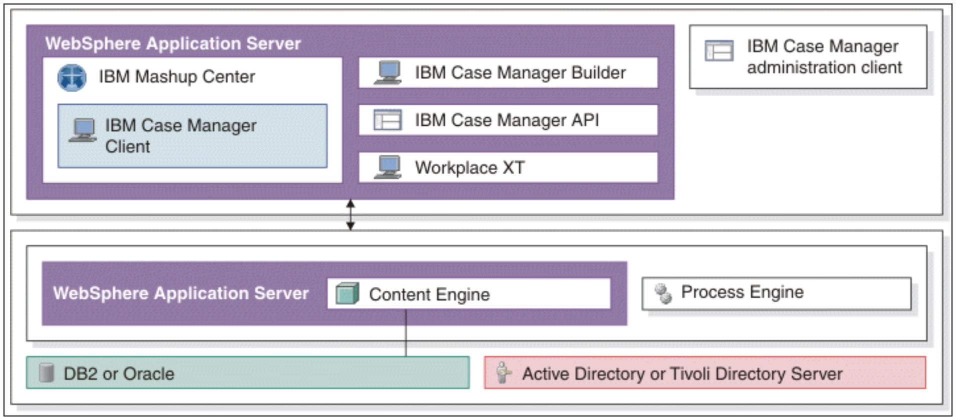

WebSphere Application Server
IBM Mashup Center
IBM Case Manager Client
IBM Case Manager Builder
IBM Case Manager API
Workplace XT
IBM Case Manager administration client
WebSphere Application Server
Content Engine
Process Engine
DB2 or Oracle
Active Directory or Tivoli Directory Server

Figure 2-13 Case Manager architecture

2.6.1 Data model

From a high-level standpoint, Case Manager uses a solution-based data model. The solution defines the task to be completed using the following items: case types, roles, pages, in-baskets, and document types. These items are implemented as a single Process Engine isolated region and application space and one or more Content Engine object stores.

Roles/In-baskets

A *role* represents a specific business function, for example, a role might be an Applicant or a Supervisor. Users are assigned to roles, and roles allow users to access a particular task.

Roles are defined within a solution and are associate roles with tasks. The Case Manager Client assigns users and groups to roles which specifies which users can access a particular task or step. Roles can be reused across all *case types* within the solution.

Case types

Case types define the tasks, the necessary document types to support the task, the task steps, and the roles that must complete those steps to solve a business problem. The case type also includes properties that are displayed to case workers in views in the Case Manager Client. Related case types make up a solution. A case is an instance of a case type.

Tasks

A case contains *tasks*. A task has one or more steps that must be completed to complete the task, for example, a task might be to review new hire applications. A case is not complete until all required tasks are completed or manually disabled. Each task has roles that are associated with it.

A task can be implemented either as a Case Builder workflow or a WebSphere component or process. Case builder tasks are implemented as a Process Engine workflow and can be round-tripped between the Case Builder and the Process Designer. For WebSphere components, it must be an empty task in Case Builder, and must be implemented using the WebSphere Integration Developer (WID).

Tasks have states to indicate where the task is in the process of completion. These states are defined as waiting, ready, working, complete, and failed, as shown in Figure 2-14.

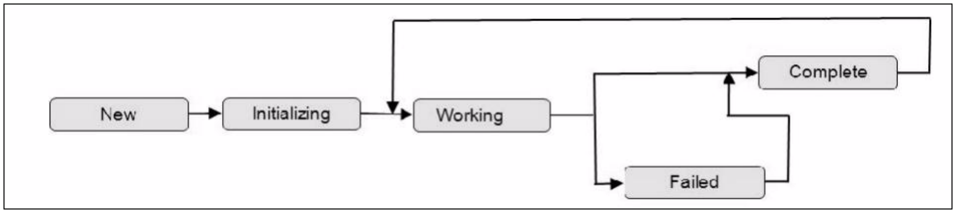


Figure 2-14 Task state transition and flow

Cases

Within Case Manager, a *case* is defined as a collection of tasks and supporting material. A case is complete when all required tasks are completed. Within the Content Engine, a case is represented as a folder. This folder contains the supporting material and objects.

Content Engine elements

A Case Manager implementation is defined by two object stores within the Content Engine: a design object store and a target object store.

Design object store

The design object store is where Case Manager stores configuration data, such as mashup layouts, pages, and solution templates. Within the design object store, each case solution is represented by a set of files, the "solution package" including the solution definition file (SDF), configuration XML files, and workflow definitions.

Finally, within Case Manager and IBM FileNet P8 domain, there is only one design object store. The Case Builder UI only reads and writes from the design object store, and the object store is configured using the Case Manager Administration Client.

Target object store

While the design object store holds the configuration, it is the target object store that actually holds the deployed solution. It is also the target object store that the case workers interact with and where case instances are stored.

The target object store can be a new or existing object store and can have one or more solutions deployed to it. A Case Manager deployment in an IBM FileNet P8 domain can support multiple target object stores. At deployment time, the CMAC uses the solution configuration to generate all required artifacts in the underlying object stores and isolated regions, including classes, folders, and workflow instances.

2.6.2 User interface

Case Manager ships with three primary user interfaces: Case Builder, Case Manager Client, and Case Manager Administration Client.

Case Builder

The case builder is the primary tool for building a case-based solution, which it does by providing tools for creating case types, defining tasks, and building pages. The Case Builder is designed so that a business analyst can build the solution using the web-based application. The Case Builder also allows the analyst to deploy the solution to a development or test environment for rapid prototyping and testing.

After a solution is deployed to the test environment, the analyst can then work sample cases and tests in a sandboxed testing area using the Case Manager Client. The solution can be returned to the Case Builder for additional refinement as necessary.

After a solution is refined and is ready to be deployed, the analyst then works with the administrators to deploy the complete solution to the production environment. This deployment occurs using the Case Manager Administration Client.

Case Manager Client

While the Case Builder is used by business analysts to define case types, the Case Manager Client is the primary user interface for case workers to process case assignments. The Case Manager Client is a widget and web-based application. Because it is widget based, it is possible to include additional functionality as part of the solution without necessarily needing to develop a custom application or user interface.

The Case Manager Client is divided into three default spaces: the solution space, step pages space, and the case pages space. These spaces all include a set of widgets that can be used directly. The spaces can also be rearranged to use custom built and other vendor's widgets.

Solution space

The Solution space is a mashup space that case workers use for interacting with a solution. It is the main IBM Case Manager space and contains two pages:

- Work page, with the In-baskets and Toolbar widgets
- Cases page, with the Case Search-related widgets

Step pages space

The step pages space is a mashup space that includes step pages that display when a case worker opens a work item. The step pages space contains three pages:

- Add task page for adding a new task for the case.
- Work details page for working on a work item.
- Work details eForm page for working on a form-based work item.

Case pages space

The case pages space is a mashup space that includes pages that display when a case worker opens a case to view its details or add a case. There are two types of case pages: those used to create a case, and those used to open and view the details of an existing case. The case pages space is divided into two sections, the case toolbar widget and the case information widget.

Case Manager Administration Client

The Case Manager Administration Client is similar to the P8 Configuration Client in that it bootstraps the various components (Case Builder, Integration Tier, and Widgets) and helps deploy them to WebSphere Application Server. The Case Manager Administration Client:

- Builds Case Builder, widgets, and P8 WebSphere Application Server "trust" (LTPA Key)
- Creates connection profiles
- Used to deploy a solution to production
- Configures Lotus Mashup spaces and pages

- Registers default solution space and case detail and work item pages in the design object store
- Used to test or “visualize” a solution without requiring the Business Analyst to create spaces or pages

Pages are discoverable from Case Builder and can be associated with a solution at design time.

2.6.3 API

Combined with the existing IBM FileNet P8 APIs, Case Manager offers an API that provides full access to case content and process. The APIs are REST-based and allow for dynamic case manipulation and creation. The API can be used to create and assign new case activities on the fly and manipulate the native case object model.

2.6.4 Case templating

The Case Manager solution also offers the ability to package a solution into a template. This allows customers and third parties to solve a specific case management problem and then package the solution into a reusable template. These templates can include:

- Industry vertical sample templates
- Pre-defined process as well as dynamic activity creation and assignment
- Run-time collaboration
- Cross-case analysis and reporting

2.6.5 WebSphere Process Server integration

Case Manager also offers strong links into the WebSphere Process Server. These links allow a Case Manager workflow to initiate or be part of a WebSphere Process Server workflow. This integration allows for leveraging existing WebSphere Process Server workflows and connections with other systems. It also allows for the addition of case management functionality to WebSphere Process Server workflows.

3

Application interfaces

In this chapter, the various interfaces available for interacting with P8 Platform are discussed. In addition to the standard application clients, two included with IBM FileNet Business Process Manager are discussed.

This chapter covers the following topics:

- 3.1, “Workplace XT” on page 64
- 3.2, “Workplace” on page 67
- 3.3, “Electronic forms” on page 68
- 3.4, “Business Process Framework” on page 74
- 3.5, “ECM Widgets” on page 82
- 3.6, “Summary” on page 89

3.1 Workplace XT

Workplace XT provides the presentation layer and includes user interfaces for the IBM FileNet P8 system that are readily available. Workplace XT is a browser-based client application that is written with the IBM FileNet APIs that are included with the Content Engine. Written in JavaServer Faces (JSF) and based upon Web 2.0 technology, such as AJAX, Workplace XT runs within a Web Container on a J2EE application server. Workplace XT provides a user interface for content-oriented tasks, such as browsing, searching, authoring, and viewing content. Other features include accessing workflow queues and processing forms (with the P8 eForms or Lotus Forms add-on). Workplace XT does not support customization.

Workplace XT consists of four views:

- Browse: A general folder browse interface for browsing and working with documents stored in folders.
- Search: Access to various search facilities for locating documents based on properties or content criteria within the system.
- Tasks: Provides views into inboxes and public queues for working on business process work items.
- My Workplace: Provides a portal view to multiple Workplace XT views in one place.

Browse view

Browsing is used to find and work with the documents, folders, and stored searches that are available on a site. The top-level container for items accessed in Workplace XT is by default the object store. The object store can contain documents, folders, stored searches, and search templates. A folder contains the items filed within it, such as subfolders or documents. A stored search displays the document or folder objects that match a search criteria.

Workplace XT also provides an option to store user favorites. Favorites are used to create references to the documents, folders, or predefined searches most frequently accessed by the user, similar to the use of Bookmarks or Favorites defined in a browser.

Search view

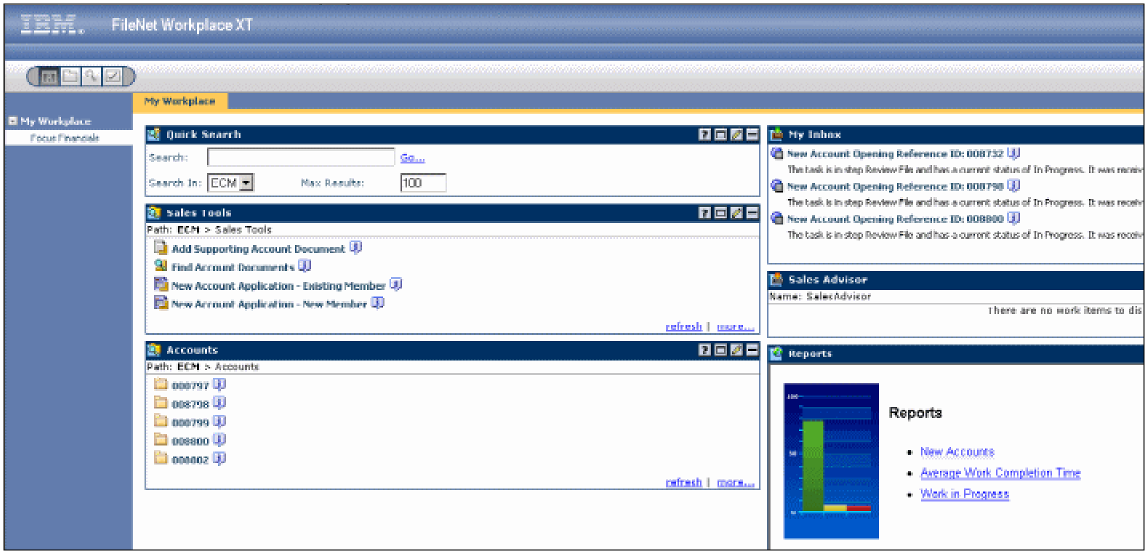
Documents or folders can be located in Workplace XT based on either keywords or their properties. Searching based on properties allows objects to be located based on the values contained in their metadata properties. The system provides the ability to perform predefined searches that are defined through the Search Designer, as well as the ability to perform ad-hoc searches through the Simple, Keyword, and Advanced Search pages. Ad-hoc searches through the Advanced Search page can also be saved on a per-user basis.

Tasks view

The Tasks mode in Workplace XT provides quick access to workflows and work items that users have permission to view or process. As workflows are started (either manually or automatically), work items are routed to queues for processing by authorized users. Users can be presented with queue and work item information within the various Tasks views, which include My Inbox view, Public Inboxes, My Active Workflows, and Task Tracker.

My Workplace view

The My Workplace page (Figure 3-1) provides a way to compose various aspects of the Workplace XT application into composite views focusing on information that is relevant to a user's daily tasks. It can also include a portlet for external Web sites, such as a corporate or industry site, for reference. For example, with My Workplace, users can view items in an inbox, view multiple folders and their content, see search results, review the list of active launched workflows, review the list of work items in a specific public inbox, and see the Author tools in a single view. Composed pages are shared among multiple users by role.



FileNet Workplace XT

My Workplace

Quick Search
Search:
Search In: ECM
Max Results: 100

Sales Tools
Path: ECM > Sales Tools
- Add Supporting Account Document
- Find Account Documents
- New Account Application - Existing Member
- New Account Application - New Member

Accounts
Path: ECM > Accounts
- 008797
- 008798
- 008799
- 008800
- 008802

My Inbox
- New Account Opening Reference ID: 008732
  The task is in step Review File and has a current status of In Progress. It was receiv
- New Account Opening Reference ID: 008798
  The task is in step Review File and has a current status of In Progress. It was receiv
- New Account Opening Reference ID: 008800
  The task is in step Review File and has a current status of In Progress. It was receiv

Sales Advisor
Name: SalesAdvisor
There are no work items to dis

Reports

Reports
- New Accounts
- Average Work Completion Time
- Work in Progress

Figure 3-1   Workplace XT

3.1.1 Tools

Workplace XT provides tools for administration, the creation of templates and searches, and business process administration and configuration.

- Search Designer: A Java applet-based tool for creating stored searches and search templates.
- Entry Template Designer: Define and edit entry templates used to streamline the process of creating document folders and forms.
- Security Policy Tool: Create and modify security templates used to control document access at different phases of its life cycle.
- Process Designer: The Java applet-based tool creates and manages workflow maps.
- Workflow Subscription Tool: Links workflows with classes to create subscriptions that automatically launch workflows on document creation events.
- Process Simulator Console: Process simulation console.
- Process Simulator Designer: Design simulations for the process simulator.
- Site Preferences: Administer the appearance, behavior, and connectivity of the FileNet P8 Platform client applications.

3.1.2 Creating content

Workplace XT offers three basic methods of document and object creation: wizard-based, a drag-and-drop target, and entry templates. The Add Document wizard prompts the user through the steps necessary to create a new document in P8 Content Manager. The drop area initiates the Add Document wizard when one or more files or folders are dragged onto it from the file system.

The third content creation method in Workplace XT involves using Entry Templates and is intended for situations where administrators must simplify options for a user when creating a document, for example, it might be necessary to allow only documents of a specific type to be created in a given folder or to hide properties from a user when entering the metadata for a document. This capability can be used to significantly improve the productivity and accuracy of document creation.

3.1.3 Viewing images

Many documents stored in P8 Content Engine will display in their associated application or viewer on the user’s client machine when opened. Workplace XT includes a viewer to display image documents and can view, zoom, magnify, scroll, pan, rotate, print, and add annotations to them. Image Viewer supports TIFF, BMP, GIF, JPEG and JPG, and COLD file formats.

By default, each of these file types opens automatically in Image Viewer. However, a site administrator can configure these file types to open in other programs, if so desired.

3.1.4 Architecture

Workplace XT runs within a web container on a J2EE application server. It uses the P8 Content Engine and Process Engine APIs.

3.1.5 Customization

Workplace XT *does not* provide a mechanism for customization through any toolkit, extensible framework or source code. It is client application only that is readily available.

Customers who require customizations must instead use Workplace, which is described in the next section. Workplace XT works with eForm and ECM Widgets, but it does not work with Business Process Framework (BPF). Workplace XT is also required if using IBM Case Manager.

3.1.6 Workplace XT summary

Workplace XT offers a simple-to-use but capability-rich user experience for content and process management. Its intuitive, role-based Web 2.0-based interface delivers access to the powerful capabilities of P8 and allows users to adopt it quickly.

3.2 Workplace

Like the Workplace XT application described in the previous section, Workplace provides access to the document management features of P8 and is tightly integrated with its business process management capabilities. Although the appearance differs in a number of significant ways from Workplace XT, many of the same capabilities exist, including content browsing, search, eForms integration, records management integration, image viewing, and the My Workplace portlet views. Workplace works with Business Process Framework whereas Workplace XT does not work with Business Process Framework. Workplace does not support ECM Widgets, but Workplace XT does.

No new features are being added to Workplace. Workplace XT, on the other hand, is being updated and offers many significant usability advantages.

3.2.1 Architecture

Workplace is deployed and run within a web container on a J2EE application server.

3.2.2 Customization

Workplace includes a variety of heritage JSP and Servlet components and is built using the IBM FileNet Web Application Toolkit, which is an extensible framework and reusable modules for building Web applications. The Toolkit supplies the behaviors and data structures for authentication, event routing, state information, preferences, localization, and other features. Organizations can modify the Workplace source code for specific purposes and needs to integrate Workplace with other systems or to use it as a basis for custom application creation.

> **Customization:** As mentioned earlier in the Workplace XT section, Workplace XT does not provide a mechanism for customization. What works with Workplace does not necessarily work with Workplace XT.

3.2.3 Workplace summary

Workplace delivers an interface for users that is readily available. Users can search and use content and interact with business processes. Although Workplace is still in use within many organizations, Workplace XT, FileNet Integration for Microsoft Office and ECM Widgets deliver current and compelling user experiences for today's ECM needs.

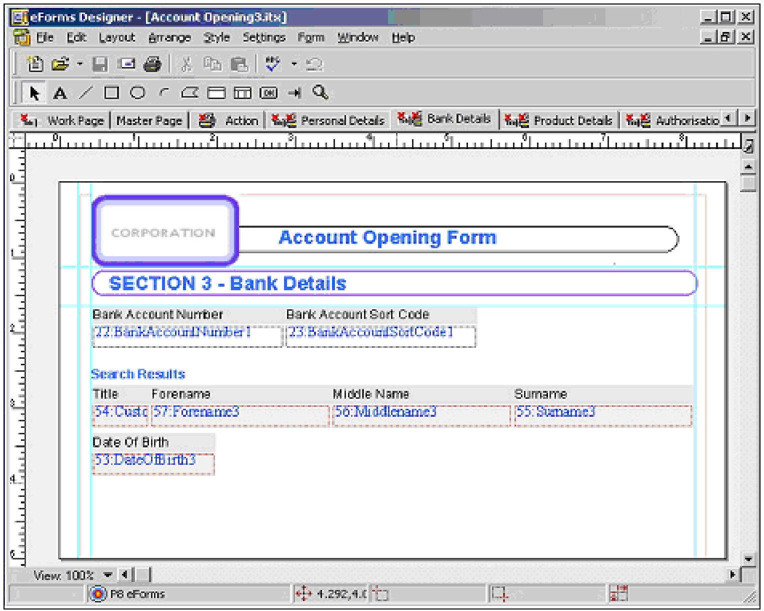
3.3 Electronic forms

Electronic forms (eForms) allow rapid development of intuitive user interfaces, enabling quick response to business needs. Electronic form products are designed to replace paper-based forms and custom web development to provide a single, flexible platform with which to rapidly develop user interfaces for business applications. The IBM FileNet P8 Platform supports electronic form technology with IBM FileNet eForms and Lotus Forms. This section discusses the architecture and use of FileNet eForms and briefly covers Lotus Forms, where appropriate.

FileNet eForms is a component of IBM FileNet Business Process Manager, and is an add-on to Workplace or Workplace XT. It includes a Designer tool and enables key functionality, such as electronic signatures, business process integration, and data lookup and validation. As part of the IBM FileNet P8 Platform, these forms can become part of business processes, automating and streamlining work and enabling businesses to quickly transform cumbersome paper forms into fully interactive electronic forms.

IBM FileNet eForms Designer tool

Business users, architects, and other form designers create and manage FileNet eForms using the IBM FileNet eForms Designer, a Windows-based, graphical tool for creating and configuring many various types of fields within a form, as shown in Figure 3-2. The form definitions are stored as form templates.



CORPORATION

Account Opening Form

SECTION 3 - Bank Details

Bank Account Number: 22:BankAccountNumber1
Bank Account Sort Code: 23:BankAccountSortCode1

Search Results

Title	Forename	Middle Name	Surname
54:Custo	57:Forename3	56:Middlename3	55:Surname3

Date Of Birth: 53:DateOfBirth3

Figure 3-2 The FileNet eForms Designer application

Editors can create forms that are electronically identical to paper forms with tight graphical control, field layout, sections, and ordering. Field content is also designed within this tool from presentation to calculation and validation.

Electronic signatures and data lookup are two key features of electronic forms. Electronic forms offer unique capabilities over their paper-based equivalents by allowing users to digitally sign areas of the form and transmit this information over a network, which is much quicker than paper-based delivery and less expensive than creating a scanning and indexing operation. Electronic forms also save on the storage of printed paper.

Electronic forms can lookup and validate values for fields in the form, for example, a customer can type in a customer number and have his or her address information automatically retrieved and populated onto the form. This feature not only serves as a convenience, but also improves accuracy and reduces opportunities for data that is incorrectly typed.

Desktop eForms is especially useful for cases that involve taking a form policy and definition offline in situations where there might not be a network connection. A good example of this is an insurance claims assessor who goes to customers and assesses the damage to a vehicle after an accident. After the assessor returns to his or her office, the completed claims forms can be uploaded through a web page in Workplace or Workplace XT. After it is synchronized with P8 Content Manager, they are brought online and processed like any other form.

3.3.1 Architecture

FileNet eForms and Lotus Forms make full use of the underlying IBM FileNet P8 Platform's ECM and BPM functionality, as shown in Figure 3-3 on page 71, which shows how Workplace or Workplace XT is responsible for determining how web-based and thick-client applications interact with form definitions and data. Note that Business Process Framework integration is only applicable when working with Workplace, and not with Workplace XT.

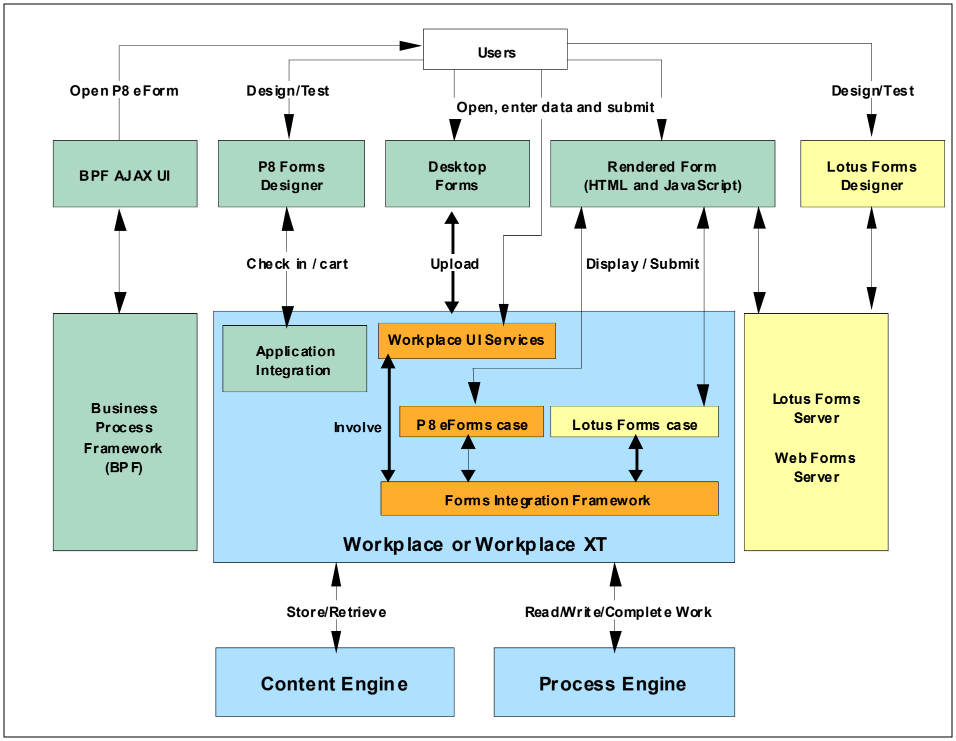


Figure 3-3 eForms architecture as supported by the IBM FileNet P8 Platform

Form templates are stored in the Content Engine like any other document. After a form is stored to the Content Engine, an administrator can create a form policy to instruct IBM FileNet P8 how to handle the data that a user provides, thus creating a form data object (a completed form). The form data object itself can be saved as a document and linked back to the original template. A business process can be launched in addition to or instead of storing the form. These form data objects and processes can use field data that is captured on the form. How that data is used and exchanged is specified in Form Policies or Workflow Subscriptions. Form data objects can be opened later and displayed exactly as they were filled in by the user, faithfully reproducing the data and its formatting.

This platform capability of linking a form template to a content repository and BPM suite was also extended to allow Lotus Forms to be used in the same way. This integration allows customers already using Lotus Forms to continue using their preferred forms solution while harnessing the power of the IBM FileNet P8 Platform.

An abstraction layer called the Forms Integration Framework handles the underlying concept of form data classes and form definition objects. When a user clicks a completed form in Workplace or Workplace XT, it is this library that determines which forms product was used to create that form. This library then invokes the relevant plug-ins to render the form. Additional configured options, such as form window size and title, Submit and Cancel buttons, and optional side panes (to show instructions, for example), are also rendered in the same window.

This standard mechanism abstracts the concept of form templates and form data objects to decouple the product from the IBM FileNet P8 Platform, which makes it easier to manage because administrators only have to learn core forms concepts, such as definition classes, data documents, and form policies, to work with either forms product.

The eForms server must be collocated with Workplace or Workplace XT in the same application server instance, sharing the same Java Virtual Machine. In WebSphere, for example, this means deploying into the same profile.

The Lotus Forms Web application provides an alternative for electronic forms, offering complex data lookups, validations, and dynamic loading of sub forms to the displayed Lotus Form. When used in conjunction with Lotus Forms, IBM FileNet P8 Content Manager serves as a form storage layer and provides an integration layer with IBM FileNet Business Process Manager. The IBM FileNet eForms library provides similar functionality of lookups and validation for any rendered IBM FileNet eForms. For either form technology, communication between the client browser and the Workplace or Workplace XT server is handled through HTTP or HTTPS.

3.3.2 Integration

eForms can be remotely invoked using a URL-based mechanism for opening a form definition for a user to populate. This URL can optionally pass initial data into the form. A return URL can be included with the request, which means that when the form is submitted into Workplace or Workplace XT, the user's browser can be sent to a specified URL with information about the unique identifier of the form data document in the system.

This capability makes plugging eForms into a custom developed application simple and shields application developers from any internal changes to the IBM FileNet P8 eForms functionality by providing a standard URL-based interface.

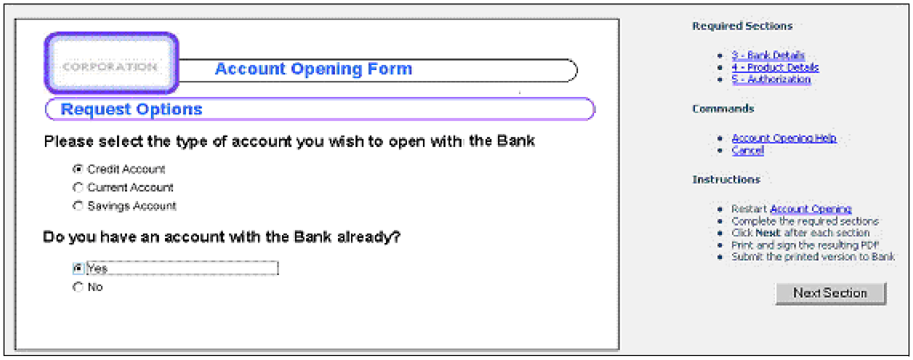
Developers who want to provide custom form-based solutions can use either the eForms JavaScript API or the eForms Java API. The FileNet P8 Form Data Java API provides developers of custom application or features with the ability to access a Form Data instance associated with a specific workflow. The API is built specifically for scenarios where organizations want to repurpose the form data generated from a running workflow and associated Workflow Policy.

The API is packaged as com.filenet.eforms.api in the following file:

```
<app_root>/WEB-INF/lib/p8eforms.jar
```

Using the IBM FileNet eForms JavaScript API developers can create unique value-add solutions. A common application of this flexibility is to use the JavaScript to create wizard-driven forms that guide the user step-by-step through a process. In addition to navigation, the JavaScript API also provides access to data at the field level, integration with on-form events, such as button clicks, and many other capabilities.

Figure 3-4 shows a navigational banner JSP page, displayed on the right side of the form, which shows the individual pages that are available in the form. Pages in the navigation can be added or omitted based on the user's interaction with the form.



Please select the type of account you wish to open with the Bank

● Credit Account
○ Current Account
○ Savings Account

Do you have an account with the Bank already?

● Yes
○ No

Required Sections

- 3 - Bank Details
- 4 - Product Details
- 5 - Authorization

Commands

- Account Opening Help
- Cancel

Instructions

- Restart Account Opening
- Complete the required sections
- Click **Next** after each section
- Print and sign the resulting PDF
- Submit the printed version to Bank

Figure 3-4 A wizard-driven form created using the eForm JavaScript APIs

This information can then be displayed in the summary page, which makes the whole user experience much more intuitive and less cluttered because the data being captured can be logically segmented into separate form pages.

IBM FileNet eForms is integrated with the Business Process Framework application when used in conjunction with Workplace. Typically in BPF, the case tab is a simple grouping of textual fields that are shown within a grid box, which can be configured to use an eForm instead of the basic case tab. This functionality creates a consistent method of user interaction throughout an application.

Thick client Windows applications, such as the FileNet eForms Designer and FileNet Desktop eForms, are supported using communication either through the Workplace .JSP files or Workplace XT Web pages.

3.3.3 Dependencies

The eForms Designer application requires a Windows client operating system. Refer to the following web location for a complete description of the eForms system requirements:

`http://www.ibm.com/support/docview.wss?uid=swg27013654&aid=5`

3.3.4 eForms summary

The IBM FileNet P8 Platform's built-in support for IBM FileNet eForms and Lotus Forms provides a quick and easily extensible method for creating a rich user interface for business applications. Electronic form support can be used to create intuitive and efficient user interfaces to support business needs. They are quick to develop, easier to maintain than custom web pages, and can be plugged into third-party applications.

IBM FileNet eForms also uses the same underlying functionality of the IBM FileNet P8 Platform to provide a standard document class and metadata model. The IBM FileNet P8 Platform also enables speedy creation of business process applications and uses eForms as a feature-rich interface with a rapid time to production.

3.4 Business Process Framework

IBM FileNet Business Process Framework (BPF) is a development framework included with IBM FileNet Business Process Manager as an add-on to Workplace (not Workplace XT). BPF reduces the time and cost of developing process-based applications by providing a configurable framework for BPM application creation. BPF is intended for building end-user applications for processes that are long-lived, requiring multiple interactions that often require review and approval.

This section discusses uses for BPF as a means of rapidly building case applications with a tool that P8 customers have used for years. If you are building a new case-based application, it is recommended that you evaluate IBM Case Manager as a preferred starting point.

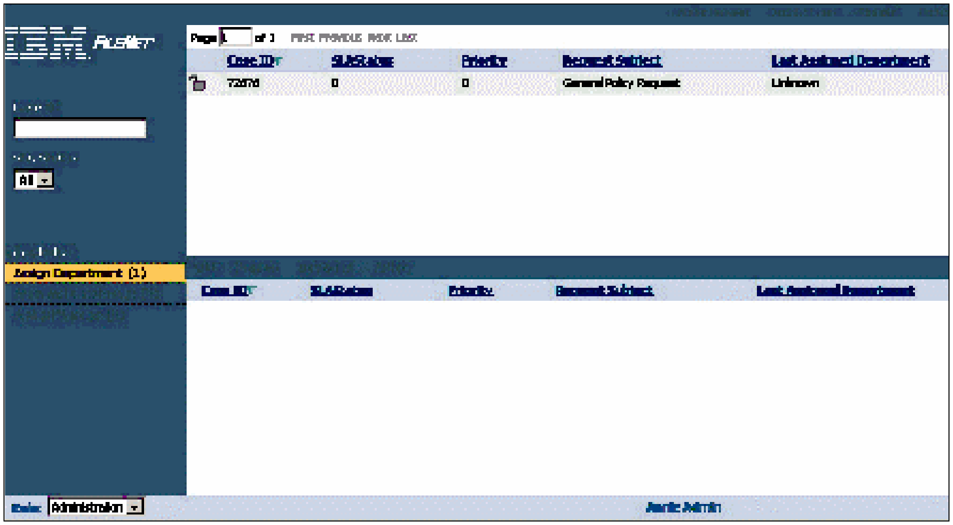
The Business Process Framework is built on top of the IBM FileNet P8 Platform, with readily available functions that can be used as is and that are tailored specifically to scenarios where case workflows, inboxes, and rapid user interface design are needed. This readily available function allows organizations to focus on configuring rather than coding, which greatly accelerates application development.

> **Application compatibility:** Business Process Framework works only with Workplace. It is *not* supported by Workplace XT.

Case

Long processes with many interactions can generate a large amount of data. These processes might contain simple values, such as a claim valuation, or more complex data, such as a detailed inspection and survey document. All of this information within the processes and content interactions must be linked together and made available for the entire life cycle of the process. In traditional, paper-based methods of managing such processes, a filing system is maintained with a case folder for each customer or transaction type. This case folder concept is useful for long-lived interactions and represents the model on which a BPF case is based.

The Business Process Framework provides this case metaphor for electronic documents and data that are part of a business process. It provides mechanisms for creating various types of cases, each of which might serve different business needs. BPF links these case objects to the relevant documents, data, processes, and folders in the ECM repository. BPF also provides a standard user interface to enable the rapid configuration of case-based applications. Figure 3-5 shows a sample case application in BPF.



	Case ID	SLAStatus	Priority	Request Subject	Last Assigned Department
	72676	0	0	General Policy Request	Unknown

Case ID	SLAStatus	Priority	Request Subject	Last Assigned Department

Figure 3-5 Sample generic case application in BPF

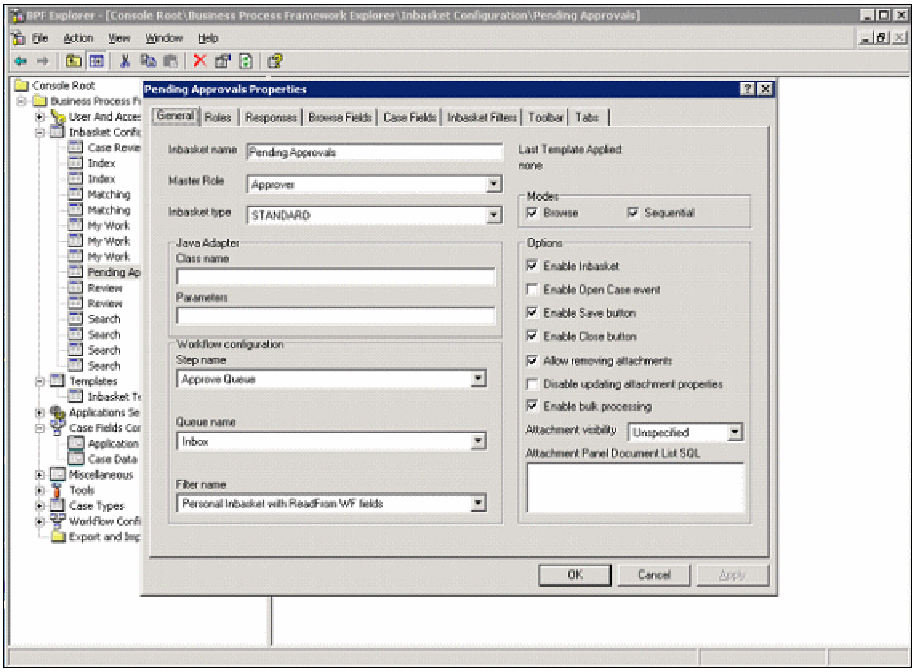
Roles and inbaskets

In Figure 3-5 on page 75, a user with the role of Administrator is shown logged into BPF. This user's role shows three available inbaskets, and cases in the currently selected inbasket are shown on the right. A user can have multiple roles that are granted to them based on their job role or group membership. Each role can have a unique user interface layout that provides the correct information to them to perform that role. Each role might also have one or more inbaskets. An inbasket is simply a view of work to be completed within the system, which extends the concept of queues by allowing the BPF designer to restrict the queue items that appear in the inbasket by providing a queue filter.

BPF Explorer

BPF also provides a configuration tool called BPF Explorer, which allows administrators to rapidly configure, rather than code, a case management application. As shown in Figure 3-6 on page 77, BPF roles, inbaskets, queue filters, case types, and lookup fields can all be configured within a single interface. The BPF Explorer is used to create an easily modifiable environment without requiring programming skills and enables real-time changes in a production environment.

In BPF Explorer, the links between the case, data, and associated business process flow are configured, providing BPF the correct responses and steps to handle in the cases. In Figure 3-6 on page 77, the properties for the inbasket for Pending Approvals are shown, and a wide variety of information about the inbasket can be configured, including roles, responses, fields available, filters, toolbars, and tabs. For more information about defining case behavior, see BPF classes and documentation.



Inbasket name: Pending Approvals

Master Role: Approver

Inbasket type: STANDARD

Java Adapter

Class name:

Parameters:

Workflow configuration

Step name: Approve Queue

Queue name: Inbox

Filter name: Personal Inbasket with ReadFrom WF fields

Last Template Applied
none

Modes

☑ Browse ☑ Sequential

Options

☑ Enable Inbasket

☐ Enable Open Case event

☑ Enable Save button

☑ Enable Close button

☑ Allow removing attachments

☐ Disable updating attachment properties

☑ Enable bulk processing

Attachment visibility: Unspecified

Attachment Panel Document List SQL

Figure 3-6 BPF Explorer with Inbasket properties

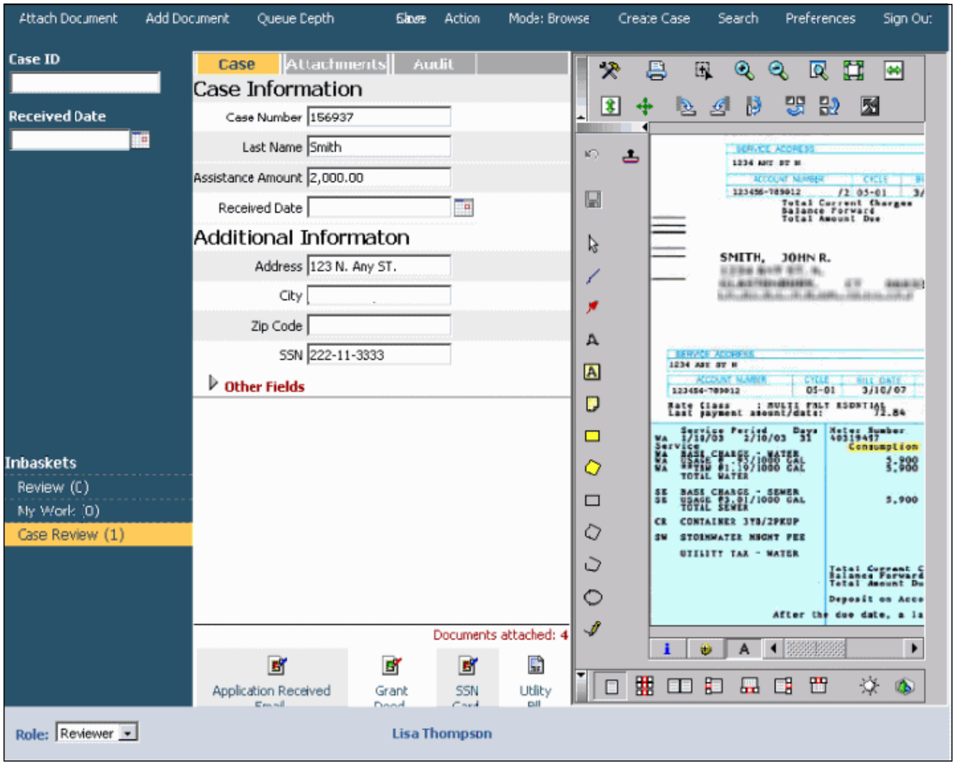
BPF Layout Designer

The BPF Layout Designer is a visual design tool that permits the application architect to replace the default layout of the BPF Web Application user interface by creating new Layout Objects. Layouts are then assigned to one or more user roles so that separate layouts can be created for each role.

In Figure 3-5 on page 75, various modules were dragged onto separate portions of the UI to create a new user interface. Many of these modules are configurable from there, such as adding the logo location address.

Case tools

Business Process Framework provides several user interface features that can be configured and reordered to rapidly create a customized case application. Figure 3-7 on page 78 shows an example.



Case ID

Received Date

Case Information

Case Number 156937

Last Name Smith

Assistance Amount 2,000.00

Received Date

Additional Informaton

Address 123 N. Any ST.

City

Zip Code

SSN 222-11-3333

Documents attached: 4

Role: Reviewer

Figure 3-7 Case management interface

As shown in Figure 3-7, the interface consists of several areas:

- Case view

  This is a tabbed area of the interface where the main data of the case is shown, which includes a Case Information tab, Attachments tab, and Audit History tabs. Custom tabs can also be added.

- Inbasket list

  The inbaskets can be shown either as a list or as a drop-down selection.

- Application toolbar

  This area provides functions for the entire application, which includes user preferences, logout designer link, and search actions.

- Case toolbar

  This toolbar provides the user with options that are applicable to either the inbasket view (when the list of work items is shown) or to the currently open case. These tools can be configured to include custom actions, such as links to eForms to populate or customization to add a comment to the case audit history for collaboration.

- Logo

  A logo element is provided to allow the organizations logo to appear on the interface.

- Roles drop-down

  A drop-down box is provided to allow the user to select which role he or she wants to see in the inbaskets and work for. Most users will have a single role, but this element supports multiple roles.

- External URL

  A component is available to show an external URL, which can be a link to best practice information, a search engine, or any other interface that is useful to the user with the current role.

- Image Viewer

  The image viewer applet displays image files directly within the BPF interface. If this is omitted from the layout design, when an image is opened a window shows the image viewer instead.

Business Process Framework also supports some additional user interface features:

- Case search inbaskets

  Allows users to search the entire content repository for BPF cases, regardless of their current inbasket or even whether or not they still have an active workflow.

- Case creation

  Allows the user to create a new case, which can include adding attachments. Creating the case begins a new business process based on the configuration for this type of case.

- eForm case tab replacement

  The case tab provides functionality to type in case data, view read only data, and restrict data choices to external lookups and drop-down options. It does not provide sophisticated calculation fields or validation. The case tab interface is a relatively simple single page of field names and values that are organized within a table. For more sophisticated user interfaces, it can be replaced by an eForm. eForms can be designed with calculation and validation fields and have a rich desktop publishing-like support for form design, which provides a rich user interface option to replace the case information tab within BPF. Other tabs, such as Attachments and Audit, can be shown as normal and are unaffected by this feature.

- Work sorting

  Queue filters can be used to provide a default ordering to work items. This sorting can be used as priority, which can be calculated within a business process or if SLA timers and escalations must be met.

- Push and pull working options

  In some situations, it is desirable to force case workers to work on the next highest priority work item rather than let them pick and choose which work item to complete next. To support this, BPF inbaskets can be configured to get the next work item. When a user opens such an inbasket, or is shown the first/next case, it depends on how the inbasket is configured to sort work items.

3.4.1 Architecture

Business Process Framework uses many of the underlying features of the IBM FileNet P8 Platform, adding only two new architectural elements, which are the BPF configuration database and the BPF administrative and design Web application, BPF Explorer. The configuration database holds information about where case objects are created within the ECM repository, which queue filters to use, and inbasket configuration. The Web application reads this database to show particular user interface elements to certain users and to allow layout design using a web interface.

Figure 3-8 shows the key architecture of the Business Process Framework.

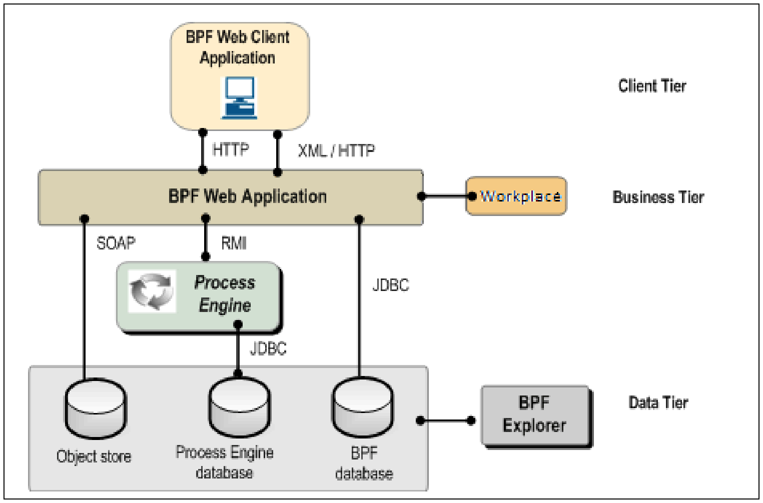


Figure 3-8 BPF architecture

All information other than the BPF configuration is held in or accessed from the ECM and BPM systems, themselves. The case objects, attachments, and audit entries are stored as custom object classes within one or more IBM FileNet Content Manager object stores. The process responses, work assignees, and queue work lists are all managed within IBM FileNet P8 BPM. BPF also provides a Java component, called BPF_Operations, that is installed into the component integrator to allow business processes to create, read, and update cases and their attachments and to add audit entries.

BPF uses Asynchronous JavaScript and XML (AJAX) techniques to load information into the Web browser, which means that instead of redrawing the entire Web page when a link is selected, BPF only requests the minimum amount of information that is required to complete its task and redraw only the parts of the interface that need changing. This functionality minimizes the amount of information that flows over the network and makes BPF perform faster than traditional web applications. AJAX interfaces are intended to make Web applications perform as fast as their desktop application counterparts.

3.4.2 Integration and connection points

Business Process Framework exposes a number of customization areas for extending its features and capabilities. Code examples for the most common customizations are available for download.

In general terms, these customization areas fall under three broad categories:

- Client-side JavaScript EventHandlers
- Server-side JSP/Java plugins
- Cosmetic changes to .CSS and .GIF files

The client-side JavaScript EventHandlers interface for BPF provides the ability to implement custom code that is run on the browser client workstation in response to a variety of events that might occur as the user interacts with BPF.

The server-side JSP/Java plug-ins consist of such features as compiled Java classes that are configured to fire when a case is opened or dispatched from a designated inbasket, custom tools and tabs, lookups, and custom preferences.

A new Cascading Style Sheet (CSS) file can be created to modify the appearance of the BPF Web Application. There is also the option to customize the look and feel of the BPF Web Application by modifying the base XSL files that BPF uses.

Developers can also make modifications to the Bp8Actions.xml file to create custom actions within the IBM FileNet BPF Case user interface. These actions are available by right-clicking attachments in the attachments pane of the BPF case view or by right-clicking an attachment in the list view pane of the BPF viewer window.

It is also possible to write custom tabs, tools, and lookups and plug them into BPF, as long as they conform to the expected interfaces.

For additional information about extending and integrating Business Process Framework, refer to the document entitled, “IBM FileNet Business Process Framework Developer Guide”.

3.4.3 BPF summary

The Business Process Framework (BPF) provides a rich set of functionality on top of the core IBM FileNet P8 Platform to rapidly design, configure, prototype, and deploy process applications. BPF provides a rich, easy-to-use user interface. BPF also provides access to information in the underlying ECM repository, business processes, and external systems and presents this as a single, updatable case object. BPF enables IT and business analysts to rapidly create business applications without the custom coding and time investments that are associated with traditional business re-engineering projects. BPF has been used to create applications that have gone from design to production in a short period of time.

3.5 ECM Widgets

IBM Enterprise Content Management Widgets consist of a collection of portable, reusable components, also called widgets, that can be used to create user interfaces for Enterprise Content Management (ECM) and Business Process Management (BPM) applications. IBM ECM Widgets enable applications to be built that expose content from both IBM FileNet P8 and IBM Content Manager Enterprise Edition content servers.

IBM ECM Widgets provide a set of standard ECM and BPM widgets based on the iWidget specification that can be rapidly integrated to create composite applications.

Unlike the low-level P8 APIs used by application developers, IBM ECM Widgets are intended for use by non-developers, allowing them to:

- Create simple applications without additional programming
- Create custom applications by integrating custom widgets from other vendors
- Rapidly incorporate business process management functionality into ECM applications

Applications built using IBM ECM Widgets are developed in a mashup environment, such as Business Space powered by WebSphere. A mashup environment makes it fast and easy for users of all skill levels to create simple, process-driven Web applications from existing information sources.

A mashup environment consists of two types of interfaces: those that enable business analysts to create and manage applications and those that facilitate the completion of work.

To create an application using IBM ECM Widgets, a business analyst performs the following tasks:

- Models workflows in IBM FileNet Process Engine workflow applications, such as Process Designer
- Creates pages and collections of pages, called *spaces*, in the mashup environment
- Drags and drops widgets on a page
- Configures the widgets to be included in the application
- Specifies which users can access a space or a page

The widgets that are included with IBM ECM Widgets can be separated into four main categories based on function. However, it is also possible to create custom widgets and to incorporate those from other vendors with IBM ECM Widgets within the same application.

Process widgets

Process widgets are used to view and process work items. IBM ECM Widgets includes the following process widgets:

- In-basket widget: Used to view the in-baskets and work items associated with a role.
- Step Processor widgets: Used to create, update, and complete work items.
- Process History widget: Used to view the status of milestones defined in a workflow.

Content widgets

Content widgets are used to view and edit documents stored on either an IBM FileNet P8 or IBM Content Manager server. IBM ECM Widgets include the following content widgets:

- Content List widget: Used to display a list of documents that are on an IBM FileNet Content Engine server and to check documents in and out, download documents, and view document properties.

- IBM WEBi widget: Used to search for and work with documents stored on an IBM Content Manager Enterprise Edition server.

Viewer widgets

The Viewer widget is used to display documents stored on a content server. The viewer that is used depends on the content server from which the document is being retrieved.

Toolbar and action widgets

Toolbar and action widgets are used to add custom actions to an IBM ECM Widgets application. IBM ECM Widgets include the following toolbar and action widgets:

- Toolbar widget: Used to select an action to perform. The actions that are available depend on how the toolbar widget is configured.
- Launch Process action widget: Used to create a work item by launching a workflow.
- Launch Process (eForm) action widget: Used to create a work item by using an eForm that is associated with the workflow.
- Open Web Page action widget: Used to open a specified Web page in a separate browser window.

Events

To establish communication between widgets, most ECM widgets publish events and handle events, for example, clicking a button on a widget might cause the widget to publish an event to send data to another widget. The widget that receives a published event has a corresponding event to handle the incoming data. The data that is published or handled by an event is called a *payload*.

The process of defining event communications between widgets is called *wiring* widgets. Many of the ECM widgets are automatically wired to the other ECM widgets on the page. As a result, when an IBM ECM Widget is added to a page, the published and handled events that support automatic wiring are connected to others without requiring any additional configuration, for example, the step processor widgets that can be added to a step processor page are automatically wired to the Step Completion widget. The Send Work Item event that is published by the Step Completion widget is automatically configured to send its payload to the Header, Work Data, and Attachment widgets. Each of these widgets contains a Receive Work Item handled event, which processes the payload received in the Send Work Item event.

The Content List widget and the WEBi widget must be manually wired to other widgets. In addition, all events that are exposed by ECM widgets can be manually wired to widgets from other sources.

Payload types

The payload types define the format of data that is passed between widgets when events are fired. The payload types that are used by the events must be known to develop or wire widgets so that they can subscribe to events or publish events. The included JavaScript Adapter widget can be used to transform non-matching payloads by wiring it between widgets.

For communications between the REST application programming interfaces (APIs) and between widgets, payloads are formatted as JavaScript Object Notation (JSON) objects.

3.5.1 Architecture

ECM Widgets are tightly integrated with the FileNet P8 Platform, requiring the back-end services of Process Engine and Content Engine and the middleware services of the Process Engine REST Service and Workplace XT. The Process Engine REST Service enables communication between web clients and the Process Engine. Workplace XT provides access to documents and other content that can be incorporated into workflows.

An IBM ECM Widgets application includes a set of widgets configured to work together to enable users to manage content and process work items.

Figure 3-9 on page 86 illustrates the basic components of an IBM ECM Widgets application.

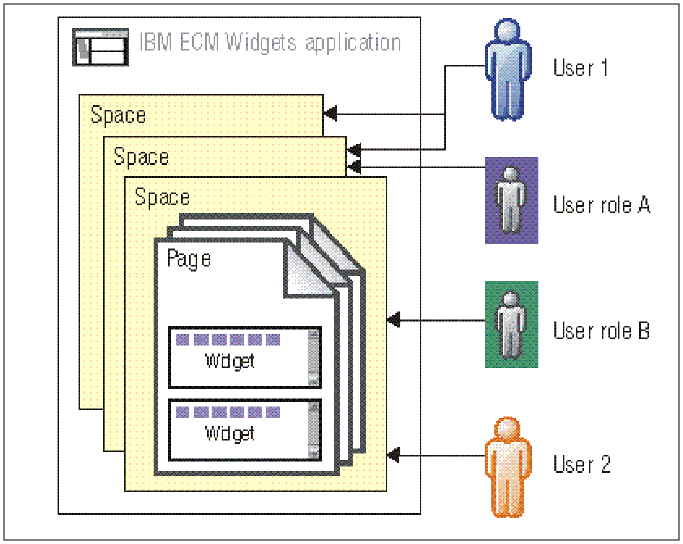


Figure 3-9 ECM Widget application components

Pages

The widgets that are required to complete a task are configured on a *page*. A page is usually associated with a particular task or set of tasks, for example, the widgets that are needed to create a new loan request are grouped on a page. If a user must complete multiple tasks, there might be a separate page for each task that must be completed.

Spaces

Pages are organized within *spaces*. Spaces are typically associated with a department or role within an organization and are used to group pages of widgets that users need to complete a set of associated tasks. An application can consist of multiple spaces, for example, an application for a bank can have spaces for home loans, auto loans, new accounts, business accounts, and so on.

A user must be authorized to access a space before he or she can open the pages in the space. Access can be assigned to individual users by specifying user IDs or to a group of users by specifying a role. When a user logs into an application, the user sees the spaces and pages to which he or she was granted access. The user can also see his or her assigned work items.

Roles

Roles and In-baskets are defined in Process Configuration Console and Process Designer and determine the privileges that a user has within a BPM application. The roles that are available to users are determined by their logon and their membership in one or more roles. The In-basket widget then displays one or more in-basket lists for the currently selected role.

Work items

A work item is a task that must be completed by a user. A work item is associated with a step in a workflow, and a user's work items are displayed in an in-basket. If a user has work items from more than one workflow, a user might need multiple in-baskets or an in-basket that displays work items from multiple workflows.

The steps in a process are displayed in the step processor associated with it, which can be a mashup page, Workplace XT, or a custom step processor.

3.5.2 Integration and connection points

Custom widgets can be integrated with IBM ECM Widgets by wiring an event that is published or handled by the custom widget to the corresponding event in the IBM ECM Widgets. To create an application with IBM ECM Widgets, a business analyst must be familiar with the following applications:

- WebSphere Application Server or WebSphere Application Server Network Deployment

  IBM ECM Widgets can be deployed on either WebSphere Application Server or WebSphere Application Server Network Deployment. However, the version of WebSphere Application Server to use depends on the mashup environment.

- A mashup environment, specifically Business Space powered by WebSphere

  Business Space powered by WebSphere is used to assemble, wire, and share an IBM ECM Widgets application.

- IBM FileNet Workplace XT

  IBM FileNet Workplace XT enables IBM ECM Widgets to communicate with IBM FileNet P8 content servers. Workplace XT is also used to create stored searches and entry templates and to open applications such as Process Designer.

- IBM FileNet Process Engine Process Designer

  IBM FileNet Process Engine Process Designer is used to model and define workflows, queues, in-baskets, and roles; to associate in-baskets with roles; and to register and associate step processor pages with workflow steps.

Depending on the other products that are to be integrated with IBM ECM Widgets, familiarity with the following applications might also be required:

- IBM Web Interface for Content Management

  IBM ECM Widgets uses IBM WEBi to communicate with Content Manager EE servers.

- IBM FileNet eForms for P8

  eForms Designer can be used to create templates for rendering step data in step processors.

3.5.3 Dependencies

IBM ECM Widgets can be installed in a container and deployed on IBM WebSphere Application Server configurations using Business Space powered by WebSphere on AIX® and Windows platforms. The IBM ECM Widgets installation program also installs the package that is used to deploy Business Space powered by WebSphere for systems where manual deployment is necessary.

The following components are needed for configurations using IBM FileNet Content Manager object stores:

- IBM WebSphere Application Server
- IBM Installation Manager: This product can be downloaded from the IBM web site at the following location:

  http://www.ibm.com/support/docview.wss?uid=swg24024682

- Database: Either DB2 for Linux, UNIX® and Windows or Oracle
- Content Engine
- Process Engine

For a complete list of supported versions of required platform software, see the IBM FileNet P8 Hardware and Software Requirements document at:

http://www.ibm.com/support/docview.wss?rs=3278&uid=swg27013654

3.5.4 ECM Widgets summary

IBM ECM Widgets offer a fast, easy way for P8 applications to be created without the need for coding. The process, content, viewer, and action widgets included with the ECM Widgets support a wide range of powerful functions for quickly assembling interfaces for applications involving enterprise content and business process management. The application also enables an IT user to create custom widgets that can be integrated with existing ECM widgets.

3.6 Summary

In this chapter, we discussed a variety of standard applications that address integrated content and process requirements. Workplace XT offers a rich set of ECM features without the need for custom coding. eForms provide an intuitive interface for form-driven processes, as well as an effective replacement for paper. Business Process Framework integrates case-based processes and their associated content into rapidly-delivered applications, while ECM Widgets offer a simple approach to building highly functional applications using a standard widgets framework.

4

Expansion products for content ingestion

IBM FileNet P8 provides a number of major expansion products that extend enterprise content management functionality beyond the core platform. Expansion products implement critical features to ingest, organize, and access content, connect information, and users expediently and support discovery and compliance while giving greater insight into critical business trends. This chapter provides an overview of the major expansion products for the P8 Platform and then focus specifically on the content-ingestion related expansion products. Other key expansion products are discussed in other chapters.

This chapter covers the following topics:

4.1 Expansion product overview

IBM FileNet P8 expansion products facilitate getting content into IBM FileNet P8, accessing content outside of IBM FileNet P8, and organizing and controlling that information. They complete the full life cycle of content from creation to storage, through usage, and finally to decommissioning. This ability to utilize existing assets in new and insightful solutions is critical for enterprise-level content management.

Expansion products covered in this book are categorized as:

- Content ingestion products:
  - IBM Content Collector
  - IBM Datacap
  - IBM FileNet Capture
- Connection and federation products:
  - IBM FileNet Services for Lotus Quickr
  - Content Management Interoperability Services
  - Content Federation Services
- Information Lifecycle Governance products:
  - IBM Enterprise Records
  - IBM Classification Module
  - IBM Content Analytics
  - eDiscovery Manager and eDiscovery Analyzer

In Chapter 4, “Expansion products for content ingestion” on page 91, content ingestion products are covered. Content ingestion solutions take paper, faxes, e-mails, and other forms of information, organize it, and insert it into IBM FileNet P8.

In Chapter 5, “Expansion products for connection and federation” on page 117, connectors and federation products are covered. Federation products use content that is already available in other repositories and locations and make it accessible and relevant. Connectors expose this information into a variety of user interfaces. Together, they increase the value of corporate assets and activate the information to participate in business processes.

In Chapter 6, “Expansion products for Information Lifecycle Governance” on page 135, Information Life Governance products are covered. It is necessary to secure, maintain, preserve, and retain the content in a useful and controlled manner. Legal requirements for document life cycles and discovery drive IBM FileNet P8 expansion products in records management and search and discovery. They expand document usefulness by annotating and classifying them for reuse. Search, classification, and discovery products organize and harvest this information.

The remaining portion of this chapter focuses on content ingestion products.

Note: Cognos Realtime Monitoring (formerly known as Cognos Now! and IBM FileNet Business Activity Monitor) is not covered in this book. See the www.ibm.com web site for support and architectural information.

4.2 Content ingestion products overview

Content ingestion products provide a solution to pull in documents from file systems, email servers, incoming faxes and images, and paper sources into IBM FileNet P8 systems. In addition to bringing existing content under control, they continue to get unexpected content under control through ongoing policy-based capture. These products not only collect this information and centralize them in the IBM FileNet P8 Platform, they also provide an abundance of other functionality.

Three key expansion products that focus on content ingestion are:

- IBM Content Collector: Expands and consolidates the previous functionality of IBM CommonStore, IBM FileNet Email Manager, IBM FileNet Records Crawler, IBM FileNet connectors for SharePoint Libraries, and IBM FileNet Application Connector for SAP R/3 (ACSAP) into a flexible software solution that collects, enhances, and manages content from Microsoft SharePoint, SAP, file shares, and email servers.
- IBM Datacap products: New to the IBM portfolio, Datacap quickly and easily captures, manages, and integrates enterprise content while extracting critical content. Datacap's offerings include easy to use customization with high-volume document capture.
- IBM FileNet Capture Professional and Advanced Document Recognition (ADR): Capture and Capture ADR are veteran products in the IBM FileNet portfolio for document capture functionality that fulfills enterprise requirements.

With these products, organizations can quickly and automatically gain control over their content. Businesses can get the right information to the right people faster with content ingestion expansion products.

4.3 IBM Content Collector

IBM Content Collector enables organizations to unlock the business value of content. Instead of merely archiving content, Content Collector breaks down silos of information by gathering and controlling it automatically. Corporations can transparently enforce compliance and operational policies while lowering the total cost of ownership. Business requirements are easily addressed with central administration, while classification and metadata makes the information more useful and reusable. Content Collector can start processes and workflows as documents are incorporated into the system. Content Collector collects, enhances, and manages content, making assets an active part in business processes.

Content Collector consists of four offerings:

- Content Collector for Email
- Content Collector for File Systems
- Content Collector for Microsoft SharePoint
- Content Collector for SAP Applications

These offerings are in the IBM ECM Content Collection and Archiving strategy line of products. Content Collector addresses four main use cases: management for compliance purposes, management for storage purposes, integration with business processes, or simply collection for eDiscovery. It is tightly integrated into the IBM FileNet P8 Platform, which simplifies and automates the process of collecting, enhancing, and managing content.

Content Collector retrieves content from these sources and applies rules to decide if and how it is processed and where they are stored. In the process, they can be classified and indexed, mined for information, removed or replaced with links to the object store, de-duplicated, and declared as records.

Although a fully detailed description of Content Collector cannot be included in this book, this section highlights those features and information related to FileNet P8 integration. Three of the offerings (Email, File Systems, and SharePoint) are almost identical in their behavior except for the source of the content and are discussed together in 4.3.1, “Overview: Email, File Systems, and SharePoint” on page 94. Differences between them are highlighted when appropriate. Content Collector for SAP Applications is covered in 4.3.2, “Content Collector for SAP applications” on page 99.

4.3.1 Overview: Email, File Systems, and SharePoint

Content Collector for Email, File Systems, and SharePoint is configured through its main application, the Content Collector Configuration Manager. This single administrative interface is used to develop and implement content collection plans and specifications. In this application, sources and destinations are identified, collections scheduled, and management and enhancement functionality specified. The following sections contain some of the terms common to this application.

Connectors

Connectors specify the source of the information to be collected and are available based on the offering installed and can include email, files, and SharePoint content. Some connectors are used for configuration for location, connection, and logging configuration for Email, File Systems, and SharePoint. There are utility connectors for functionality, such as metadata and text extraction too. A connector can specify a source, a target, or a utility.

Source connectors retrieve content and metadata but can also set metadata on source content, enabling it to stub email, for instance, or mark content as collected.

Source connectors include services for:

- Lotus Domino®
- Microsoft Exchange
- File Systems
- SharePoint
- and other connectors

Targets might have various search and metadata capabilities, and Content Collector exposes these features where possible in the Configuration Manager.

Target connectors include support for:

- Image Services
- FileNet P8
- IBM Content Manager
- File Systems

A single Content Collector server can connect to multiple types of sources. All of these services use the same APIs as part of the modular architecture.

Task connectors include:

- Classification
- Text Extraction
- Records Declaration

File Systems

Content Collector for File Systems collects files from any file system that can be mapped to a Microsoft Windows file system. In the newest version, users are no longer required to logon to ECM client applications to view stubbed documents

Email

Email is archived in a format that supports legal discovery, storage management, and duplication management. The email connector supports such features as connecting to the source system, collecting the email documents, and even a life cycle for managing the document stubbing policies. The connector also supports a flexible set of collection targets including individual mailboxes or users, all mailboxes in a group, PST files, and even local Lotus Notes® NSF archive databases.

SharePoint

The Microsoft SharePoint connector allows organizations to collect and manage content from SharePoint document libraries, picture libraries, blog libraries, and wiki libraries. As the content is collected it can be classified, organized, and even managed as important business records. The SharePoint connector can be configured to collect content from individual sites or all content from all sites in a site collection. The connector can also further filter content out by specific folder paths or by content type.

Content classification

IBM Content Collector also supports an integration with IBM Classification Module. This integration allows organizations to analyze the full textual content of a document to make more intelligent collection decisions. This integration can be used to determine how to classify the data, what type of record to declare, and can even extract additional metadata from the full text of the document.

Records declaration

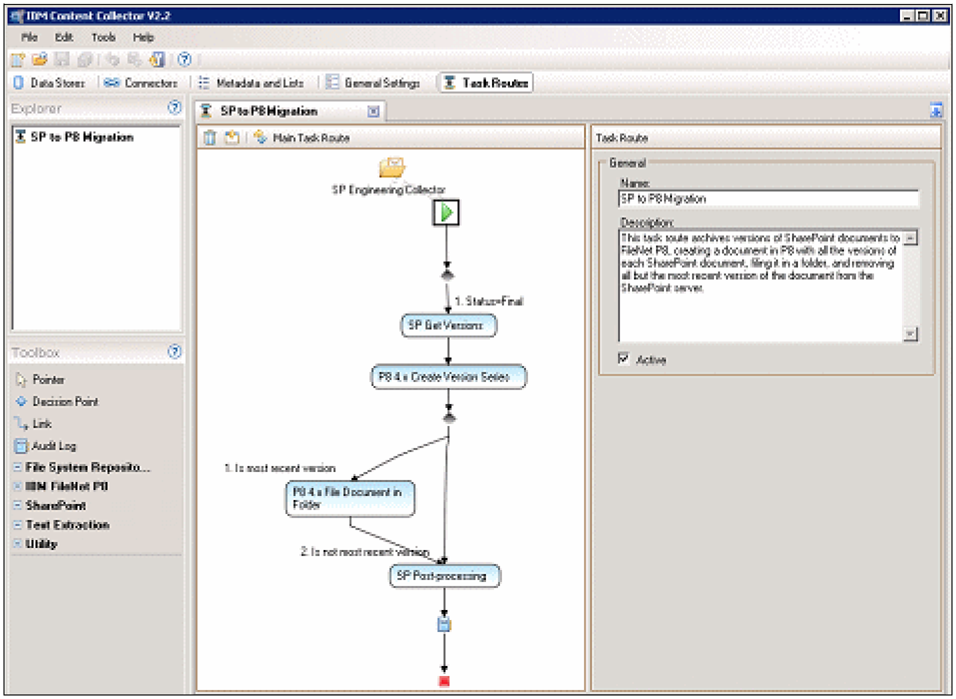
In addition to utilizing Classification Module to classify the collected content, Content Collector integrates with IBM Enterprise Records to support automatic declaration of email and files as records during the content capture process, which protects assets automatically and preserves them for future use.

Content Collector Configuration Manager

Content Collector provides a Configuration Manager for all Content Collector administration, including designing *task routes*. The Content Collector Configuration Manager is also the administration interface for all configuration that is required for the Content Collector Task Routing Engine. Other administrative tasks for repositories are managed using their respective administration tools.

Task routes

A task route is like a workflow, a visual representation of the process for content (such as emails, attachments, or files) in the system. Task routes specify how and where the content is collected and processed within the system. It begins with collection and ends with storage in the repository. Task routes apply rules at multiple points in the capture process. Decision points allow conditional processing of content using rules with potentially different outcomes for each decision. Processing assets can include extracting metadata, classification, extracting content, de-duplication, and records declaration. Figure 4-1 shows a sample task route.



Name:
SP to P8 Migration

Description:
This task route archives versions of SharePoint documents to FileNet P8, creating a document in P8 with all the versions of each SharePoint document, filing it in a folder, and removing all but the most recent version of the document from the SharePoint server.

☑ Active

Figure 4-1 Content Collector Configuration Manager with a sample task route

Connectors and task routes are the building blocks for consolidating content and breaking down information silos.

System health statistics

Key operations statistics are available for task, target, and source connectors and include documents processed, archived, and errors. This data can be viewed in Windows Performance Monitor, IBM System Director, and other tools.

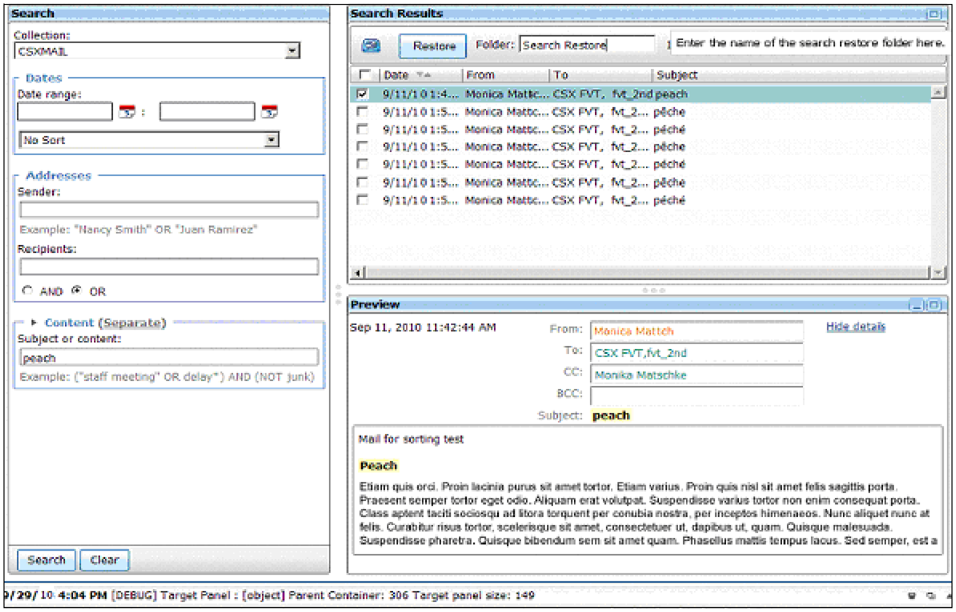
Retention Manager and Set Expiration task

The Retention Manager sweeps expired documents out of the repository, if they have no other holds placed on them. The Set Expiration task is used in task routes to set this field.

Content Collector Web application

The Content Collector Web application is used for viewing emails and searching emails in a web browser. The actions can be triggered directly from the users' email clients and the email authentication controls security. Workplace XT can use the email viewer application too.

In this web application, the user searches against their collection and can preview the results. Search texts are highlighted in the results window. Figure 4-2 shows the user interface for the email search Web application.



Collection: CSXMAIL

No Sort

Example: "Nancy Smith" OR "Juan Ramirez"

○ AND ● OR

Subject or content: peach

Example: ("staff meeting" OR delay*) AND (NOT junk)

Folder: Search Restore

Enter the name of the search restore folder here.

□	Date	From	To	Subject
☑	9/11/1 0 1:4...	Monica Mattc...	CSX FVT, fvt_2nd	peach
□	9/11/1 0 1:5...	Monica Mattc...	CSX FVT, fvt_2...	pêche
□	9/11/1 0 1:5...	Monica Mattc...	CSX FVT, fvt_2...	péché
□	9/11/1 0 1:5...	Monica Mattc...	CSX FVT, fvt_2...	pêche
□	9/11/1 0 1:5...	Monica Mattc...	CSX FVT, fvt_2...	péché
□	9/11/1 0 1:5...	Monica Mattc...	CSX FVT, fvt_2...	pêche
□	9/11/1 0 1:5...	Monica Mattc...	CSX FVT, fvt_2...	péché

Sep 11, 2010 11:42:44 AM

From: Monica Mattch

To: CSX FVT,fvt_2nd

CC: Monika Matschke

BCC:

Subject: peach

Mail for sorting test

Peach

Etiam quis orci. Proin lacinia purus sit amet tortor. Etiam varius. Proin quis nisl sit amet felis sagittis porta. Praesent semper tortor eget odio. Aliquam erat volutpat. Suspendisse varius tortor non enim consequat porta. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Nunc aliquet nunc at felis. Curabitur risus tortor, scelerisque sit amet, consectetuer ut, dapibus ut, quam. Quisque malesuada. Suspendisse pharetra. Quisque bibendum sem sit amet quam. Phasellus mattis tempus lacus. Sed semper, est a

```
9/29/10 4:04 PM [DEBUG] Target Panel : [object] Parent Container: 306 Target panel size: 149
```

Figure 4-2 Content Collector email search Web application

Application integration

Content Collector can be integrated into Lotus Notes, Lotus iNotes®, Microsoft Outlook, and Microsoft Outlook web clients. The optional outlook extension for Microsoft Outlook and Lotus Notes template modifications are available and can be installed on users' desktop computers.

4.3.2 Content Collector for SAP applications

Content Collector for SAP has a specific architecture that enhances the SAP business infrastructure with archival and portability support. SAP systems tend to produce significant volumes of data, and Content Collector for SAP archives these assets efficiently. Documents are still accessible by SAP while reducing storage costs, increasing productivity, and improving system performance.

Content Collector for SAP enables retention periods and holds on archived SAP transactions, automating compliance with regulatory and legal requirements. It includes components to archive and view data, documents, and print lists, and to link to archived documents. Multiple SAP servers can be accessed, and Image Services documents can also be utilized through FileNet P8 when CFS is installed.

Architecture

Figure 4-3 shows the Collector Server interfacing between SAP and the repositories. Three clients—the Viewing, Archiving, and Utility Clients—provide the ability to see, share, link, archive, and preserve indexing of content.

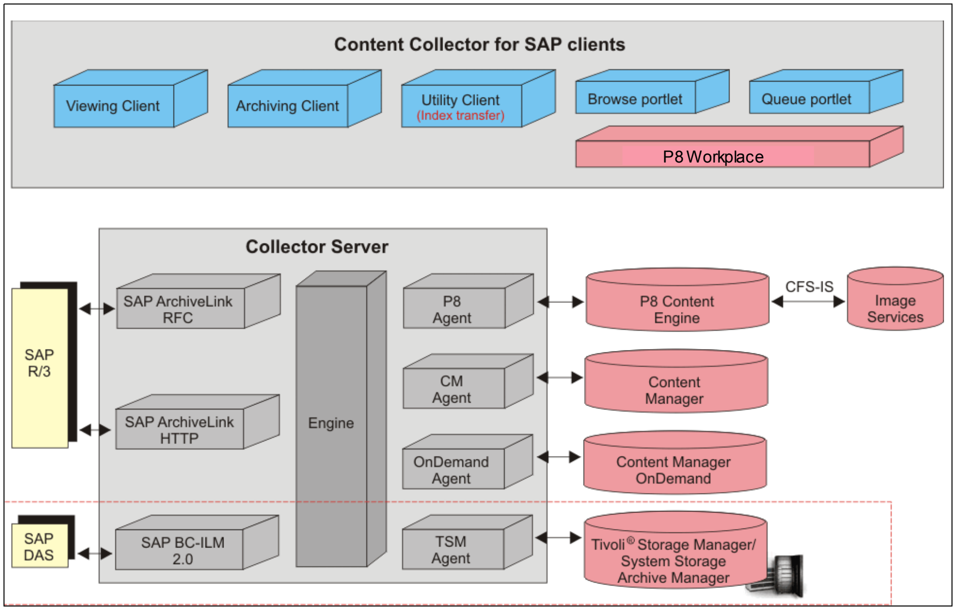


Figure 4-3 Content Collector for SAP architecture

Archiving Client

The Archiving Client is Windows based and is used to archive scanned documents. It can take documents from file systems, scanning application description files, and IBM Content Management work lists.

It supports early, simultaneous, and late (with barcode) archiving scenarios into FileNet repositories. In Early Archiving, a document is captured into a FileNet repository and made available for linking before the associated SAP business object is created. It is also the first stage of any incoming document scenario where the document is processed using SAP Business Workflow.

In Simultaneous Archiving, all document entry and SAP object processing steps are carried out by the same SAP user. Overall the process is the same as in Early Archiving except that the SAP work object, which is created at link time, is assigned to the current user.

In Late Archiving, the creation and processing of the SAP business object comes first and linking to the corresponding supporting document happens later in the process. In practical terms, this process is like a traditional paper-based process. This process can include bar code linking.

Viewing Client

The Viewing Client, also a Windows-based application, opens an external viewer for the SAP request. Users can add notes and save and share them with the document. Documents can also be emailed along with any attached notes.

Utility Client

The Utility Client can also link documents by creating a work item in an SAP workflow or by sending barcodes. It provides index transfers, which becomes metadata on the document in FileNet and searchable. FileNet documents, folders, search templates, and searches can all be linked to SAP. The Utility Client links all documents in a P8 queue. The P8 Client portlet allows selection of specific documents and versions.

P8 Client: Browse and queue portlets

The Browse portlet in P8 Workplace and Workplace XT enables linking to specific documents, stored searches templates, and entire folders. It also specifies the desired version, which creates a work item in the SAP Inbox.

The Queue portlet links documents available in a P8 queue like the Utility Client does. The work item is moved from the P8 to the SAP Inbox. For both, the user can specify which inbound linking process to use. The P8 client also supports mass archiving.

4.3.3 Content Collector summary

Content Collector products enable organizations to take back control and unlock business value of content while enforcing compliance and operational policies and lowering the total cost of ownership. With automation and centralized administration, compliance requirements become easy to address. Using classification and metadata makes the information more useful, and reusable, which enables businesses to extract more information out of their content. Content Collector can start processes and workflows as email, and files are incorporated into the system.

Content Collector provides a modular, extensible architecture for collecting content from multiple sources and applying flexible rules to the disposition of the content. The assets become organized, useful, enhanced, and managed. Businesses reap the benefits of IBM FileNet P8 rapidly, providing increased organizational agility, lowered costs, and better compliance.

4.4 IBM Datacap

Datacap is one of the IBM new acquisitions, and completes the end-to-end document management solution. Datacap's document capture and integration products help green initiatives by reducing paper requirements. Datacap captures, manages, and automates business information, and then integrates it tightly with the p8 platform, creating streamlined enterprise solutions.

4.4.1 Introduction to Datacap

The Datacap product line consists of three main products:

- Taskmaster

  Taskmaster is an SOA capture and automation solution with both web and thick clients. It is a document ingestion product.

- Rulerunner

  The Rulerunner service is a core differentiator for Datacap products and is embedded in Taskmaster and FastDoc. It can also run decoupled and can run as a web service. Easily extensible, it drives business rules and actions which make Datacap solutions so flexible and powerful.

- FastDoc Capture

  FastDoc Capture is a stand-alone product that integrates out of the box with file systems and SharePoint. It can also be operated offline. It is a document ingestion product.

Building on these core products, Datacap also offers a number of industry specific and technology specific solutions. These are discussed briefly in the products section in this chapter.

4.4.2 IBM Datacap Capture process overview

The Datacap capture process, shown in Figure 4-4, follows four main stages, although it is easily customized to follow any flow required for business solutions. These stages: scan, recognize, verify, and export cover the process of getting images, cleaning them up, mining them for information and data correction, and placing them in appropriate repositories for further use. The Rulerunner Service passes batches of images from state to state according to the business rules configured.

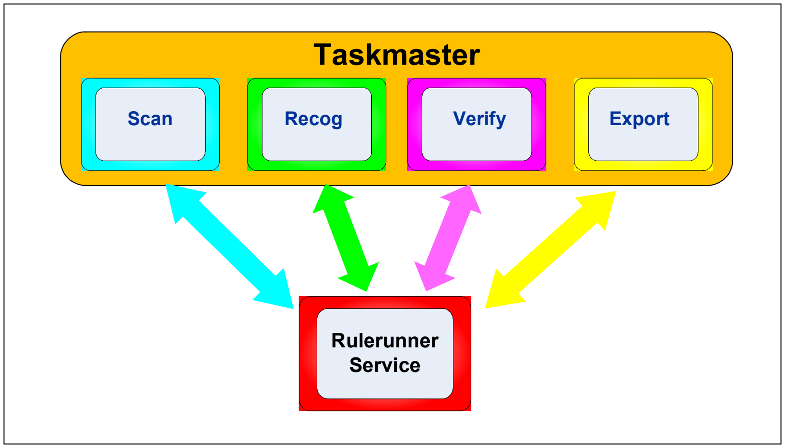


Figure 4-4 Datacap Capture process

Scan stage

The scan stage creates batch images to be processed. These images can be from a scanner, file folder, or other location. This might also include identifying the beginning and end of documents and adjusting orientation, but these might also occur in Validation.

Recognize stage

In validation, documents are processed as much as it is possible to do so automatically. Batches are separated into documents, if necessary and the batch, document, and page information are saved. Documents are scanned for all information, which is stored in a related file. This unique process means that all of the recognized text is available to users and processes in later stages, even if they were not initially identified as important. Documents are also identified according to type using *fingerprints*, which is an image file with associated data used to identify documents. Think of it like a fuzzy outline of a document's shape.

Within the document, fields for that document type are located by zone or by keyword. Barcodes, check marks, handwriting, and voting type information is also located. This information is also saved. Additional processing, such as data lookups, formatting and range validation, and other logic, can occur during this step.

There are two ways to define the location of index fields: with keywords or by Zone:

- Zone: Indicates where FastDoc Capture must look for the index value on the page. After that particular image is exported successfully, FastDoc Capture saves a fingerprint and automatically recognizes the image and the fields the next time.
- Keywords: Defines labels (sets of recognized characters or barcodes) FastDoc Capture will search for on the image, and define the position of the index value relative to that label.

Verify stage

If documents in a batch require a user's assessment and correction, those documents proceed to the Verify stage. In this stage, the user is presented with the image along with the fields and with snippets of the image as recognized in the previous stage. Users can update data and re-run validation and business logic, run other business rules, and do any final processing required.

Ideally, the Verify step leaves few, or no, images left and only uses human intervention when necessary.

Export stage

In the final stage, export tasks are performed. This stage formats and exports images and related data to p8 or other repositories.

4.4.3 Architecture

Figure 4-5 on page 104 shows a variety of ways that Datacap can support data capture and processing. Images can come in through email, fax, traditional scanners, and multi function devices (MFD). These images can enter the system through the Taskmaster web client, the thick client, directly using email and fax servers, and through network folders.

The Taskmaster server is the core server, handing out tasks with batches of images to Rulerunner stations to process. The Taskmaster Web Server hosts the web site and manages information from Taskmaster.The IBM Datacap Taskmaster Capture RV2 web application displays information about the current status of batches and other relevant activities.

After processing is complete, the images are stored along with the information gathered and generated in line of business systems, databases, and repositories such as FileNet P8.

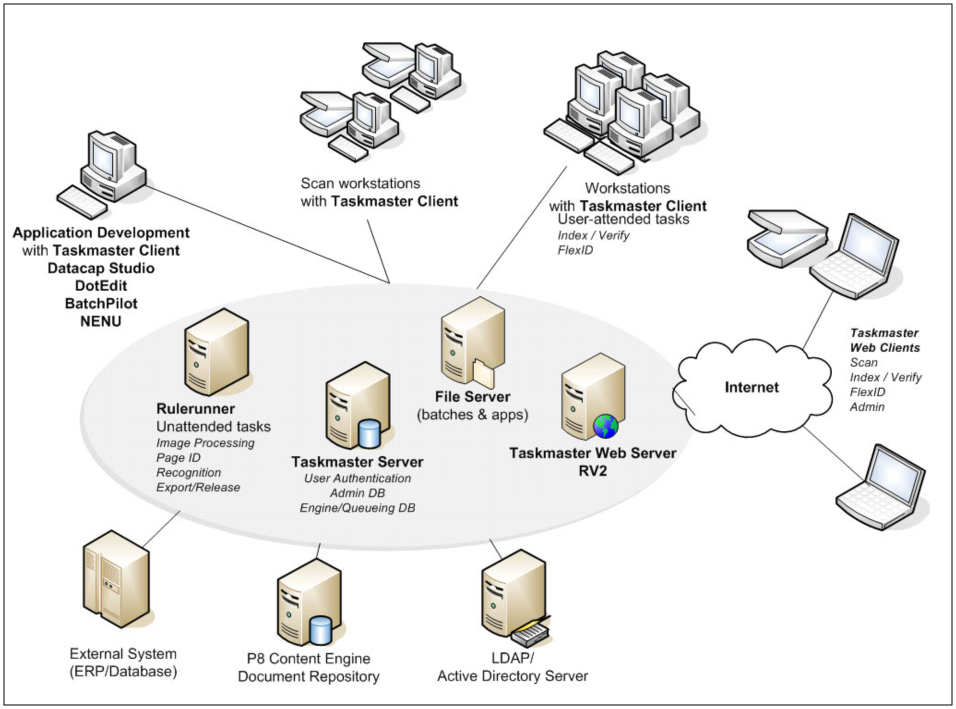


Figure 4-5 Datacap solution architecture

4.4.4 Components of the Datacap Taskmaster Capture solution

Figure 4-5 shows several key components that form the core Datacap Taskmaster Capture solution:

- Taskmaster Clients: Both thick and web clients provide the user interface. Typically this is used for validation and scanning interaction.
- Taskmaster Server: Includes authentication and permissions functionality and manages the queue of batches and jobs.
- Taskmaster Web Server: Serves pages and handles requests for web clients. The Taskmaster Web Server is only needed if the web clients are in use or if the RV2 report generator is being used.

- Rulerunner Service: Performs the actions specified on the objects provided. Its service-oriented architecture (SOA) enables it to be used by other applications. For instance, FileNet's workflows can call into the Rulerunner service to perform various recognition actions on documents already in the repository.
- Datacap Studio: Thick client tool that is used to configure objects, variables, the recognition to be used for documents and fields, and to define and configure actions to perform during processing.

Other components include tools to configure display screens and forms, manage fingerprints, and other useful functionality. The FastDoc Capture solution encapsulates the functionality of the clients, server, web server, and Rulerunner in one client.

Rules

Rules make Datacap's data processing superior because of their flexibility and reusability. Rules are code snippets that can be combined and extended easily. Users can quickly make changes and improvements because the rules are configured, rather than coded. These rules contain decades of user experience and cover a wide range of requirements and scenarios, making application creation easy. Rulesets can easily be extended in VBScript, .Net, and other languages and scripts.

A rule is typically several rulesets of several fairly simple actions, such as opening a connection to a database, performing a lookup, and then setting a field to that value. Rules perform recognition, image processing, verification, export and most other operations required by the application.

Task Profiles and jobs

Jobs are like steps in a workflow, and Task Profiles are which rules are performed during that job and in what order. An example job might be Validate, and one of the tasks Validate performs is Recognition, which applies a series of rules for cleaning up images and then running OCR on them.

Rulerunner takes a batch and applies the task profile requested on that batch. Each Rulerunner server runs a copy of the SOA rules engine, is configured for that tasks it will process, such as export or verification, and executes the set of business rules given to it according to its profile. Typically Rulerunner servers are allocated separate tasks to address load balancing. The tasks are typically split when machine types that work on that task are different, for instance, scanning is separate from profiling because profiling can run in the background.

Datacap objects

Datacap object information is stored in a batch Datacap document hierarchy (DCO), which is the document hierarchy. This is a key differentiator for the IT staff that is in charge of configuration because this information is clear, easy to understand, and manageable in several consistent places in the user interface (UI). The DCO contains information about batches, documents, pages, and fields. Figure 4-6 shows how the hierarchy fits together.

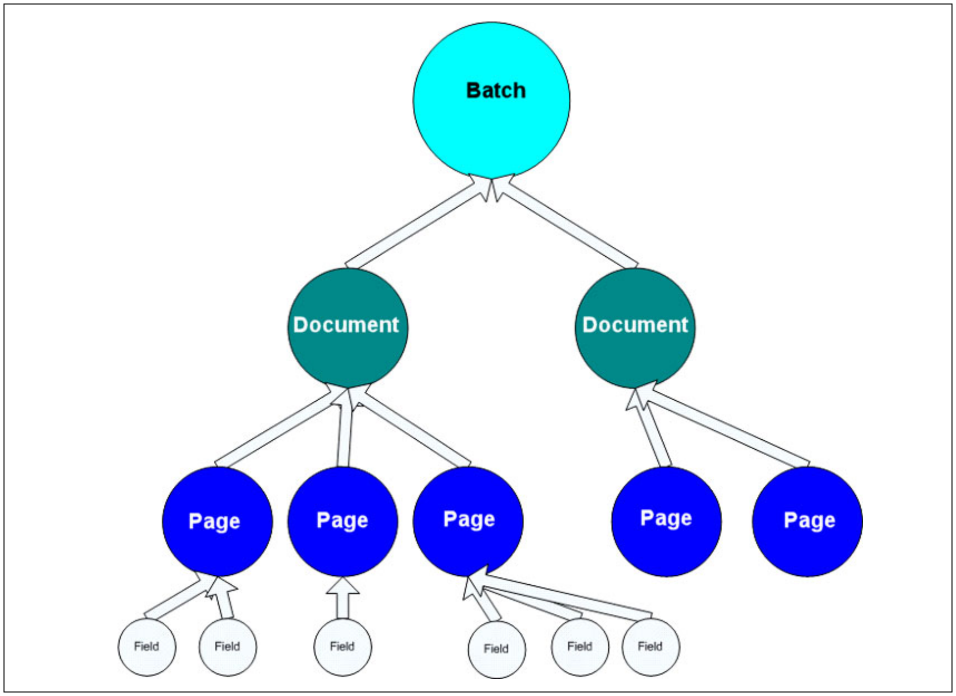


Figure 4-6 Datacap document hierarchy

Rulerunner services process rules based on the information about batches and rules. The rules are processed as defined by customized workflows in order from the batch all the way down to the fields.

4.4.5 Products and applications

Rulerunner is available as a separate product. Both Taskmaster and FastDoc Capture contain Rulerunner. FastDoc offers a rapid on-ramp for smaller or stand-alone installations, and Taskmaster offers larger enterprise-scale capabilities.

IBM Datacap FastDoc Capture

FastDoc Capture is a stand-alone thick client user interface to a Datacap Capture solution. It is used to scan, recognize, validate, verify, and export batches of images and documents offline or online and can export to files or to SharePoint natively.

FastDoc Capture has a single, intuitive interface for all activities. Administration and operation all occur within this one application. Its streamlined configuration accelerates time to production without complicated templates or programming. Automatic data recognition frees operators to focus on key issues rather than constant keying of data.

FastDoc Capture includes both user and administrative capabilities. An administrative login uses it to configure document recognition.

Administrator mode is used to:

- Set up Document Types and fields.
- Zone specific fields on the Document Type. These zones provide the Recognition file with information about the location of the zoned fields.
- Assign Keyword located fields.
- Create field validation rules.
- Set field export configuration.
- Set document export configuration.
- Perform all user capabilities.

Operator (user) mode is used to:

- Select batch sources (printers, file shares, and so on)
- Set up batches of images
- Define, order, and separate images in a batch
- Validate or correct document type classification
- Validate or correct recognized field data

4.4.6 IBM Datacap Taskmaster Capture

Datacap Taskmaster Capture uses thick and web clients to communicate with the Taskmaster Server or the Taskmaster Web server respectively. This offering has much richer functionality and comes with a variety of tools that are used to configure user interfaces, objects, and rules. There are two foundation applications built on Taskmaster Capture that are often used as base applications for creating customized solutions, saving organizations development time and harnessing Datacap's extensive experience.

Taskmaster Accounts Payable Technology

The Accounts Payable Technology (APT) application was developed for invoice processing, but its key strength is in finding data on highly variable forms. It excels at finding data that is similar from form-to-form, such as addresses or totals, whose locations move from form-to-form. Basic and advanced invoice form processing is included in this application to remove manual entry. Invoice information can also be captured using *Click and Key*, where information recognized on the form can be selected by a click to speed processing. Line items are recognized and processed regardless of the shape of the form.

Taskmaster for medical claims

This application handles United States (US) medical forms and exports it in a HIPAA-Compliant EDI stream. Professional claim forms (CMS 1500) and institutional claim forms (UB04) which are used extensively by doctors, hospitals, and insurance agencies, are pre-configured.

Sample applications

Datacap also provides sample applications that are useful as a base for creating custom solutions. This feature includes tax forms, handwriting recognition, surveys, and database-driven indexing.

4.4.7 Dependencies

Datacap solutions operate on Windows operating systems and can use Microsoft SQL server or Oracle databases for a back end. Although Datacap has a built in authentication, it can also integrate with Windows Active Directory. It supports both ISIS and TWAIN drivers; however, the scanner interfaces are not included. At the time of writing this book, English is the only supported user interface language.

4.4.8 Connection and integration points

Datacap can obtain images directly through scanners, from monitoring network drives and through controlling scanners using user interfaces. Although a full discussion of input and output image formats is contained in the product documentation, it is useful to note that it supports natively or with conversion TIFF, JPG, PDF, and some PNG formats. It exports to TIFF, JPG, PDF, PDF/A, and PNG formats.

Additionally, Datacap can interface with virtually any business application or data store through the included or customized rules and actions. Database and web services actions are often provided by Datacap. The integration with FileNet is through actions that the user needs merely to customize with specific logon information for their data store. The FileNet actions include the ability to set document type, location, and metadata.

In turn, FileNet can invoke Rulerunner as an SOA service. In this manner, document processing and other business rules can be embedded in a business process, for instance, images added to an ad hoc process can have OCR applied, barcodes scanned, and that information used to link or drive other documents and processes.

4.4.9 Datacap summary

Datacap Capture products integrate seamlessly with FileNet processes and content management. Using Datacap to ingest products helps organizations to organize content, harvest data, and control documents. Businesses streamline and accelerate the flow of information by obtaining it from the point of origin.

4.5 IBM FileNet Capture

IBM FileNet enables ingesting paper and image-based content with another set of products: IBM FileNet Capture Desktop and IBM FileNet Capture Professional. Like Datacap solutions, IBM FileNet Capture products automate control and classification during the capture process, which enhances IBM FileNet P8 compliance by increasing accuracy and lowering the risk of lost or inaccessible information. IBM FileNet Capture functionality is mostly a subset of Datacap's offerings, with three key differences:

- IBM FileNet Capture has internationalization and localization supported, particularly Asian language support for OCR in version 5.2.1.
- IBM FileNet Capture does not include advanced data extraction capabilities. The OCR in FileNet Capture is limited to simple document properties or full text.
- IBM FileNet Capture does not include many advanced features contained in Datacap capture products including web-based scanning and indexing, handwriting recognition, complex data validation, and data export.

IBM FileNet Capture Desktop and Professional can be used to scan, index, and convert content and store it in IBM FileNet P8.

4.5.1 FileNet Capture process overview

FileNet Capture processes convert paper documents into digital documents that are a representation of the original paper. There are six steps in the typical capture process:

1. Create images using scan or file import.
2. Process document, which involves image clean up and bar code/patch code recognition (Optional).
3. Acquire metadata, either through barcode recognition, zonal OCR, or manual data entry.
4. Convert to PDF (Optional).
5. Control document with Record activator (Optional).
6. Commit to IBM FileNet P8 Content Engine.

Figure 4-7 shows the basic capture functionality provided by IBM FileNet Capture. The Scan module has document processing and image cleanup already incorporated, thus scanned documents go directly to indexing. The indexing function acquires metadata from the documents. OCR2PDF is PDF conversion, which is an optional feature. Additionally, documents can be declared as records before the images are committed to the IBM FileNet P8 repository.

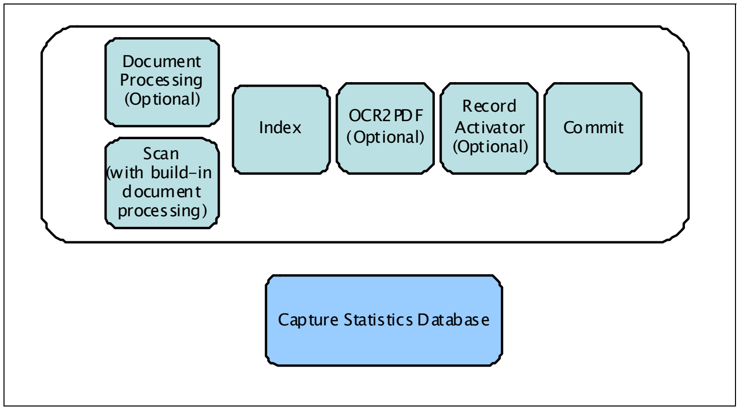

Document Processing (Optional)
Scan (with build-in document processing)
Index
OCR2PDF (Optional)
Record Activator (Optional)
Commit
Capture Statistics Database

Figure 4-7 Basic FileNet Capture functionality

Typically, paper-based content that requires capture is external to an organization, such as mailed correspondence, invoices, or technical information. This information serves to initiate, support, and further a business process. Much of the capture for paper-based documentation is centralized as an adjunct or extension to a mail room operation. Centralized capture or scan operations typically run in a specialized production environment. The need to quickly move information through such an operation requires multiple levels of expertise. There is also a high degree of validation and control to ensure that the paper-based information is moved correctly to a digital form.

Because of the steps needed to move information from paper to digital, capture supports a simplified queue system that allows batches of images to be automatically routed through the capture process. This simplified queue system is called a Capture Path, which is covered in “Capture Path” on page 112.

These capabilities constitute the essence of capture supported by IBM FileNet Capture technology and provide the ability to tailor a capture solution to meet the changing needs and specific requirements of an enterprise. All document capture components, which includes assembly, document entry, document processing, file import, and Optical Character Recognition (OCR) can be easily included or removed from the application.

4.5.2 Capture systems architecture

Figure 4-8 shows the FileNet Capture in a distributed architecture and the capture system elements. Distributed architecture is preferred for high-volume applications. Each of the steps in the capture process can be performed on separate systems.

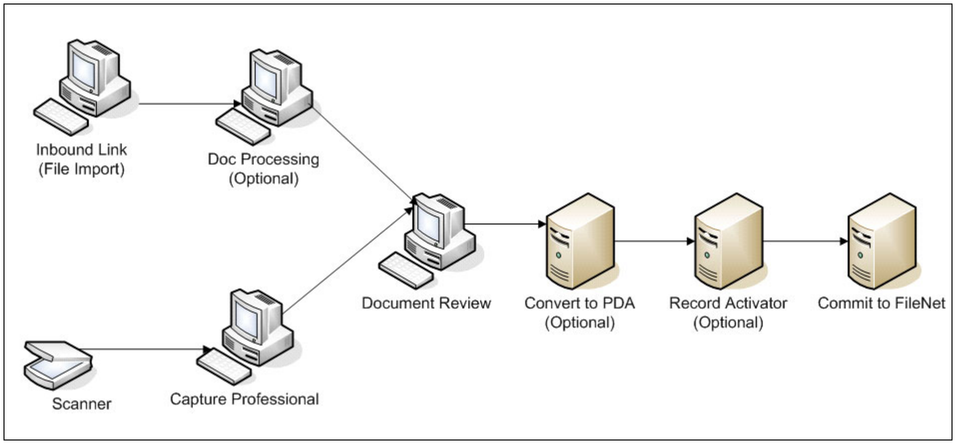


Figure 4-8 IBM FileNet Capture in a distributed architecture

Separate individual systems perform file import and optionally document processing before the capture process goes through document review. The images can also be converted to PDF and records controlled before committing to the FileNet repository.

4.5.3 IBM FileNet Capture products overview

Capture technology is an extremely critical part of any successful ECM project. Basic capture functionality is supported by IBM FileNet Capture Professional and Capture Desktop. IBM FileNet Capture Professional also includes capture paths for automation, OCR and Patch/Bar code recognition, and manual indexing.

Capture Professional includes:

- Basic capture, scan, and import
- Capture path automation
- Zonal OCR
- Patch/Bar code recognition
- Manual indexing
- Conversion to PDF
- Record declaration
- Committal to FileNet Content Engine

FileNet Capture Desktop performs scanning, indexing, and PDF conversion only and is a simplified capture solution. This subset of Capture Professional is non-distributed and contains no fax or OCR support.

The FileNet capture modules for desktop and professional support the entire range of batch capture functions and a collection of drivers for production-level scanners and the major driver standards, including:

- ISIS
- Twain
- Kofax

The next sections detail how these products expand IBM FileNet functionality, focusing on Capture Professional.

4.5.4 IBM FileNet Capture Professional functions

IBM FileNet Capture Professional provides enterprise-level production. Capture can be automated and scripted through the Capture Path, Batch Template, Settings Collection, and Capture Toolkit.

Capture Path

The *Capture Path* is a key concept in IBM FileNet Capture Professional. It defines an automated sequence of document ingestion operations to process the batch. The ability to configure and manage capture paths supports flexibility and efficiency.

Batch Template

A *Batch Template* determines what is done to a set of documents, and where they go. It is created by selecting a Settings Collection and a Capture Path.

Settings collection

A *Settings Collection* holds configuration information that defines how Capture components behave when they process a batch. In addition, the Settings Collection specifies the FileNet Repository Document Class.

Capture Toolkit

The Capture Toolkit is a part of Capture Professional and Capture Desktop that provides a rich set of sample applications, documentation, and other files that are used to develop custom Capture applications using the Capture components. Sample applications are provided with separate user interfaces for scanning or indexing, which includes document assembly, repository administration, local and multi-station automation, conversion from other systems, continuous scanning, error management, and custom components.

The toolkit takes advantage of the Capture architecture to automate document entry through built-in tools, such as Capture Paths and through custom implementations. The underlying COM objects give complete control over the user interface and Capture operations to manipulate repository servers and repository objects.

4.5.5 IBM FileNet Capture Professional components

IBM FileNet Capture Professional components support document ingestion operations that can be invoked while capturing documents. These components include Image Verify, Document Processing, Blank Page Detection, Patch/Bar code recognition, Event Activator, Assembly, OCR, Index, Index Verify, Merge, OCR2PDF, and Records Activator.

Image Verify

Image verification is used to display captured images to fix image quality and page organization. Image verification normally occurs before assembly but can also occur before assembled documents are committed. Pages can be reviewed, rejected, or marked for rescanning.

Document Processing

Document Processing (DocProcessing) provides a set of components to automate indexing and to improve image quality after scanning, faxing, or importing. This functionality also includes image clean up and Bar code/Patch Code recognition. It is optional.

Patch/Bar code recognition

Patch codes are commonly used to separate batches, and bar codes separate documents. The bar code value can exist on a separator page. Capture can interface with scanners and scanner drivers that can perform patch code recognition.

Blank Page Detection

This component detects and removes blank pages in batches.

Event Activator

The Event Activator component performs three types of actions based on rules:

- Separating objects into folders, batches, and documents.
- Changing the name that is assigned to a folder, batch, and document.
- Switching to another settings collection: supported when Event Activator is used with a batch separator rule.

Assembly

Document assembly is the process of sorting, organizing, and grouping individual pages into documents for subsequent indexing and committal. A batch is usually assembled only one time and in one of three ways: manually, ad hoc, or using a capture path.

Optical Character Recognition

Optical Character Recognition (OCR), converts parts of or all of scanned pages of machine print into editable text. Converting locations in scanned pages is known as Zonal OCR. The values that are found can become metadata or attributes of the document. The metadata can be used to route or index automatically.

Index

Indexing with IBM FileNet Capture is a coordinated process that uses index fields from the IBM FileNet server and settings, metadata, and index fields from Capture. Indexing is typically done late in the process to ensure the maximum attributes are available for indexing.

Index Verify

Index Verify is a way to double-check selected index entries before a document is committed for Image Services only. The fields that are used for Index Verify are set up on the Image Services server at the same time that indexing for the document class is set up. Normally, Index Verify is used any time after a document is indexed or auto-indexed, but before it is committed.

Merge

The Merge component combines multiple individual image files of the same or compatible type into a single multi-page file.

OCR2PDF

The OCR2PDF module performs OCR on images and generates a PDF file with embedded text that allows full text search to be performed through IBM FileNet Content Manager's search engine.

Records Activator

Records Activator provides the capability to automatically assign records management-related information of a document based on a default value for the document class, batch, or document. Documents can be associated with a specific file plan based on their attributes, such as barcode value or state.

4.5.6 Integration points

IBM FileNet Capture Professional is built using the Microsoft OLE Automation technology, which provides an object-oriented component-based architecture. This architecture allows third-party components to interact seamlessly with Capture. The Capture modular architecture allows capture solution to be tailored to meet specific enterprise needs. IBM FileNet Capture is integrated with all FileNet Repositories all through the capture process, including:

- IBM FileNet P8
- IBM FileNet Image Services
- IBM FileNet Content Services
- IBM Enterprise Records

IBM FileNet Capture uses the authentication method for the FileNet repository to which it is connecting. FileNet Capture performs real time lookup of document class and field definitions that are configured in the FileNet repository. FileNet Capture uses the FileNet repository's APIs to store documents and metadata. VBScript functions can be used to manipulate the data that FileNet Capture recognizes. These functions are also integrated with all FileNet Repositories throughout the Capture process.

Components expose Enterprise Records File Plans for record declaration and retention, which allows documents to be declared as records in the capture path, so capture works directly with Enterprise Records. Organizations can bring scanned images under records control immediately.

4.5.7 IBM FileNet Capture summary

The IBM FileNet Capture expansion product set focuses on providing a management framework that allows customers to exploit capture technologies to improve business operations. In particular, these products allow for efficient implementation of a production capture environment to quickly capture, organize, control, and utilize their documents.

4.6 Summary

Content ingestion expansion products provide core applications to quickly, efficiently, and intelligently ingest documents into IBM FileNet P8. These products not only add content but also annotate and organize the information to make it more useful. Content Collector, IBM Datacap, and IBM FileNet Capture products and expand the IBM FileNet P8 Platform add key metadata, index, and declare records while faxing, scanning, and importing files from critical business applications. These products make automation and integration simple and powerful simultaneously. Businesses gain greater knowledge and control over their mission critical information while increasing their agility in responding to market changes.

5

Expansion products for connection and federation

This chapter discusses connection and federation products that make content centrally available and usable from disparate sources, which enables businesses to maximize existing resources while exploiting the power of the IBM FileNet P8 Platform.

This chapter covers the following topics:

For an overview of all of the IBM FileNet P8 expansion products, refer to 4.1, “Expansion product overview” on page 92.

Note: The functionality of IBM FileNet Connectors for Microsoft SharePoint and IBM FileNet Application Connector for SAP Applications were rolled into the following products respectively:

- IBM Content Collector for Microsoft SharePoint
- IBM Content Collector for SAP Applications

See 4.3, “IBM Content Collector” on page 94 for additional information.

5.1 Connection and federation products overview

Connection and federation expansion products for the IBM FileNet P8 Platform protect and extend corporate investments in heritage file and data repositories. Businesses can maximize these resources by using them in new and intelligent ways without being forced to move or duplicate them. Content from databases, file storage, and team collaboration tools are now centrally available and usable assets through connection and federation products.

Teamwork and collaboration products, such as Microsoft SharePoint and Lotus Quickr, and business management solutions, such as SAP, store content in their own repositories. Connection and integration products can import, copy, or move and link documents into the P8 system and expose P8 documents directly to SharePoint and Quickr users. Connectors and services activate these documents with business processes, protect them with records management, and make them more useful and reusable with indexing and business intelligence. Connectors and services lower total maintenance costs and improve performance while maintaining availability.

Federation products manipulate business assets in place. Users can continue to use existing applications in addition to creating new solutions. The business can then implement records management, business process management, and business process optimization. In this way, corporate investments are preserved, protected, and expanded. IBM Content Integrator and IBM Content Federation Services are two examples of federation products.

In addition to these products, Content Management Interoperability Services (CMIS) is a technology that is used to federate documents using industry standards. IBM embraces the CMIS architecture, so that products by Microsoft, SAP, Oracle, and other vendors can be incorporated using one simplified, modular, industry-standard method.

Businesses break down walls of communication by ensuring connectivity between repositories. They mine new information from previously inaccessible data and control documents to meet compliance requirements. Information becomes an active, integral part of business processes. IBM FileNet P8 connection and federation products help businesses make better decisions, faster.

IBM FileNet Connectors for Microsoft SharePoint Web Parts

IBM FileNet Connectors for SharePoint Web Parts is not covered in this book. However, the Web Parts provide an excellent example of how the .Net API can be used to create direct access to FileNet Content Manager through Microsoft SharePoint. Another example of creating direct user interface access is covered in the Content Management Interoperability Services (CMIS) section.

To learn more about developing using APIs and CMIS, refer to:

- IBM developerWorks®

  `http://www.ibm.com/developerworks`

- IBM Redbooks publication

  Developing Applications with IBM FileNet P8 APIs, SG24-7743-00

5.2 IBM FileNet Services for Lotus Quickr

Lotus Quickr and IBM FileNet Services for Lotus Quickr combine collaborative authoring and sharing of everyday business content with structure, business process management, records management rules, classification and discovery. IBM FileNet P8 and Lotus Quickr unlock enterprise content and make it accessible across the corporation.

Lotus Quickr facilitates teamwork. Quickr includes customizable web sites with team places to meet, collaborate, and share information. Team places are web sites that make it easy to share information. They are customizable and configurable to meet each team's requirements with items, such as calendars, announcements, to-do lists, blogs, RSS feeds, libraries, and other useful tools.

There is also a connector integration enabling collaboration in other applications. Quickr Connectors unite applications, such as Microsoft Office, Lotus Symphony™, Microsoft Outlook, Lotus Sametime, Lotus Notes, and Windows Explorer, with Lotus Quickr with a seamless, easy-to-use, integrated interface. This union enables content sharing and collaboration in a transparent user experience, without changing applications. Lotus Quickr Connectors integrate directly with the desktop and various applications and allow users to interact with FileNet repositories as though they were any other folder. Connectors allow for both import, export, versioning, metadata, and other access to content in FileNet repository.

The Quickr user interface with ECM products results in rich application integration. Adding IBM FileNet Services to the Quickr software gives teams workflows and business process management, centralized control of content and content types, and better scalability.

Integration with IBM FileNet P8 combines collaboration with Information Lifecycle and Governance functionality, such as archiving documents to meet compliance requirements. IBM FileNet expansion products, such as IBM FileNet Records Management and e-Discovery extensions, are available and help to preserve, protect, and facilitate reuse of Quickr documents.

Together, the IBM FileNet Services for Lotus Quickr leverages the easy-to-use interface and team collaboration of Lotus Quickr and the power of active content and business process management of ECM.

5.2.1 Architecture

The IBM FileNet Services for Lotus Quickr are delivered in two parts: Lotus Quickr Server and the services themselves. The services are web applications deployed in WebSphere Application Server.

Figure 5-1 shows IBM FileNet Services and IBM Content Manager Services for Lotus Quickr. IBM FileNet Services and FileNet P8 can be deployed in the same Application Server. For best practices, they must have separate instances. Specifically, the services can be on the same server as Workplace but preferably on a separate server from the Content Engine. The Lotus Quickr box on top of the Services for Lotus Quickr represents the Quickr Connectors.

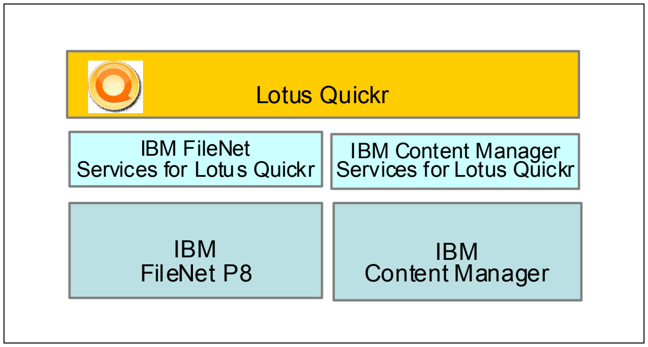

Lotus Quickr
IBM FileNet Services for Lotus Quickr
IBM Content Manager Services for Lotus Quickr
IBM FileNet P8
IBM Content Manager

Figure 5-1 Lotus Quickr modules

See documentation at the following web site for current information about version compatibility:

http://www.ibm.com/support

System architecture

Desktop applications, such as Windows Explorer, Microsoft Office, Lotus Notes, Sametime, and Symphony use Quickr Connectors, which in turn makes REST and Web Services calls to communicate with IBM ECM Content Manager Services, as shown in Figure 5-2. All of the content of these repositories are presented to the applications in the same manner. In this way, interaction with data from any source is treated the same as any other.

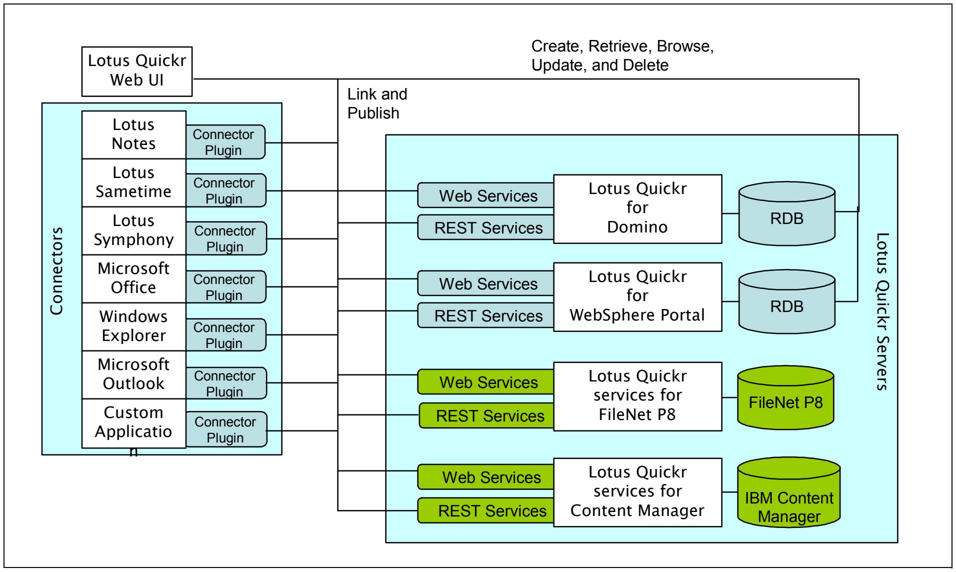


Figure 5-2 Services for Lotus Quickr connect using the Lotus Quickr connectors

The Quickr web user interface communicates directly to the database (RDB) for most actions. Link, Search, and Publish actions use the web and REST services.

Figure 5-3 on page 122 shows a more detailed architectural diagram where the delivery vehicle (the ear file) contains the services implementation. IBM FileNet P8 uses Java APIs to communicate with the services, using EJB as the transport layer.

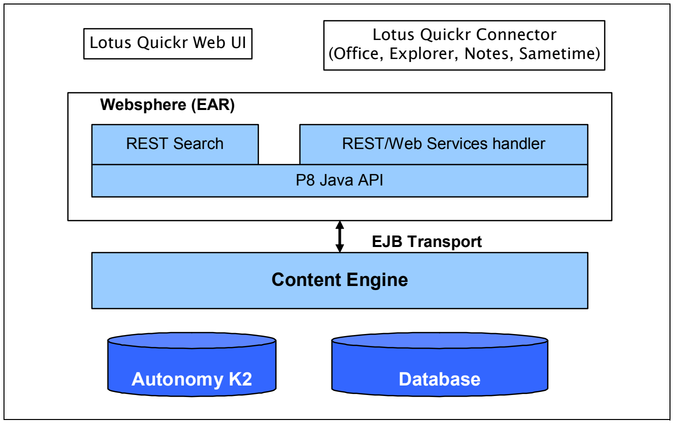


Figure 5-3 Quickr REST connections

5.2.2 User interfaces

There are two main integration points. For the Quickr web user interface, FileNet functionality is accessed through Quickr web pages using appropriate browsers. Quickr Connectors access FileNet using configured team places.

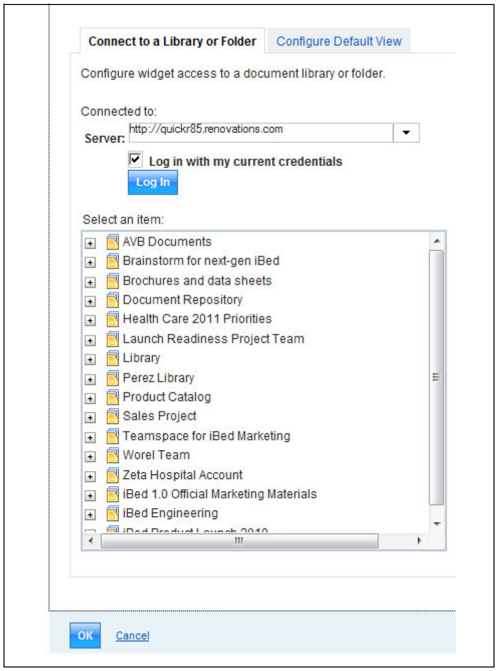
Quickr web user interface

The Quickr web user interface with IBM FileNet Services for Lotus Quickr connects to FileNet in three ways. One way to connect is through the Publish command (move, copy, and link), another is through search, and last through the Custom Library. The Publish command and search capabilities are available in Quickr libraries when configured on the Quickr and FileNet servers. The library itself is configured when added to a web page into Quickr Places, which are team web sites.

The Publish command allows users to choose to publish to a configured FileNet repository. Search over FileNet documents can now select properties. Searches and views can be saved. Administrators can create saved searches as views and share them in the web interface.

A Library is many things in Quickr: the Quickr version of a repository, the name of a page that views it and the library on that page that views it. There are two similarly named libraries: the Library, and the Custom Library. A Library views Quickr stored documents, and a Custom Library views FileNet or CM documents. The Custom Library is a direct connection to ECM repositories. The user can create, view, and update content and metadata and navigate through the Library. Documents can be checked in and out, and items and folders can be deleted.

To add a Custom Library to a Quickr Place, the user must log in as a manager or higher level role. Using the customization widget or advanced customization, they can choose the Custom Library. After it is added to a page, the manager or administrator can then configure it to use the appropriate repository location, as shown in Figure 5-4.



Configure widget access to a document library or folder.

Connected to:

Server: http://quickr85.renovations.com

☑ Log in with my current credentials

Select an item:

- AVB Documents
- Brainstorm for next-gen iBed
- Brochures and data sheets
- Document Repository
- Health Care 2011 Priorities
- Launch Readiness Project Team
- Library
- Perez Library
- Product Catalog
- Sales Project
- Teamspace for iBed Marketing
- Worel Team
- Zeta Hospital Account
- iBed 1.0 Official Marketing Materials
- iBed Engineering
- iBed Product Launch 2010

Figure 5-4 The Document Picker dialog box

The correct URL, port (if needed), and user ID and password combination are entered. Then the contents of the location are selected. When publishing from Quickr Places, it can be configured to prompt for metadata.

A similar dialog is used to select locations for publishing and linking for blogs and wikis. In a Quickr Library, a user who has publishing rights can select the Publish action for a document. They can also choose the method of publication: move, copy, or link. Linking means that the content is moved into the FileNet repository, and the Quickr Library has a link directly to that document. All three of these selections allow for FileNet to take action on that content with workflows or records management. When publishing from Quickr Places, it can be configured to prompt for metadata.

In the Search user interface, the user can choose a scope to search over from a list that is configured when search is configured. This scope tells the Search tool what repository location to search over, such as a particular folder in a particular repository.

Existing content in IBM FileNet P8 can be linked to and made available in Lotus Quickr Web user interface. The Quickr Custom Library shows a folder as a place (or a folder in a place) in Quickr Web user interface.

5.2.3 Integration and connection points

Lotus Quickr connectors provide desktop integration (direct access) to IBM FileNet P8 and IBM Content Manager (CM8) content through Lotus Quickr. This integration gives users the ability to take advantage of IBM FileNet repository and business process capabilities regardless of the environment they are in: email (Notes and Outlook), documents (Lotus Symphony and Microsoft Word), or Sametime instant messaging.

On the desktop, content can be dragged and dropped into Quickr Places to add them into the IBM FileNet repository. The user is prompted to publish the document or save as draft. Draft means that it is visible only to the owner who is editing the document. If configured, Quickr prompts the user to enter metadata specified by FileNet. Document type can also be specified. The connectors can view, create, and restore versions of documents. Properties, that is the metadata of an item, can also be viewed, added, and modified. Figure 5-5 shows the seamless integration of Lotus Quickr connectors.

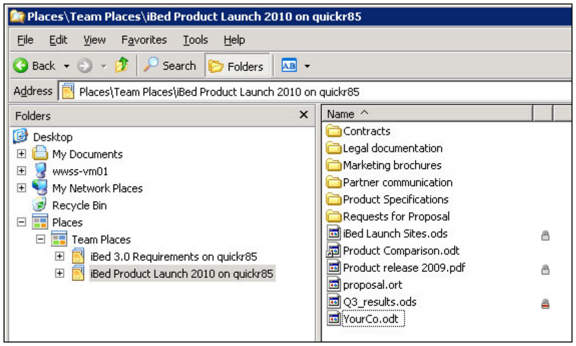


Figure 5-5 Lotus Quickr connectors integrate seamlessly

For Sametime, the same source selection dialog is available. Content can be browsed and selected in the same manner for linking.

Protocols

The library portal view in the web UI uses HTTP calls to the Lotus Quickr Server, which in turn uses REST and Web Services to connect to the IBM FileNet

Services for Lotus Quickr. The IBM FileNet Services for Lotus Quickr use EJB to communicate to FileNet for publication.

Desktop applications, such as Windows Explorer, Microsoft Office, Lotus Notes, Sametime, and Symphony, use Quickr Connectors, which in turn makes REST and Web Services calls to communicate with IBM FileNet Services for Lotus Quickr. See 5.2.1, “Architecture” on page 120. The REST services are ATOM based.

5.2.4 IBM FileNet Services for Lotus Quickr summary

IBM FileNet Services for Lotus Quickr provide an intuitive collaboration environment with business process integration. Work can be accelerated and optimized, and corporate assets managed and protected. All of these features can be accomplished without requiring additional effort or training on the part of the user. Together, Lotus Quickr and the IBM FileNet P8 Platform create an environment where teams work together to accomplish more, faster.

5.3 Content Management Interoperability Services

Content Management Interoperability Services (CMIS) standard is an approved OASIS standard, which is an industry-wide specification that supports a new openness and flexibility in enterprise content management. It is a standard that provides both a Web service and a REST binding for integration and interacting with enterprise content management systems. All participating companies (including Microsoft, EMC, Alfresco, OpenText, SAP, and Oracle) have prototypes or delivered product supporting and proving this specification.

Companies often have multiple content management systems. This is a way to seamlessly communicate with them in a standard way. ECM applications can use CMIS generically with multiple repositories, rather than more specifically integrated. The CMIS specification unlocks the business value of content because it treats various content management repositories similarly, simplifying coding and maintenance. For instance, the SAP and IBM FileNet ECM prototype of this implementation makes the integration between the two systems deeper with minimal customization and configuration.

CMIS does not replace existing interfaces; instead, it enables companies to do rapid application development with a least common denominator set of interfaces.

5.3.1 Architecture

CMIS supports SOAP and Representational State Transfer (REST/Atom) based on Atom/APP and web services (WS) bindings. Every application using CMIS communicates in the same manner to every repository that supports CMIS. Each repository creates a CMIS implementation to respond to these requests.

Figure 5-6 shows how FileNet's architecture allows the platform to support other repositories easily in the future, such as IBM Content Manager.

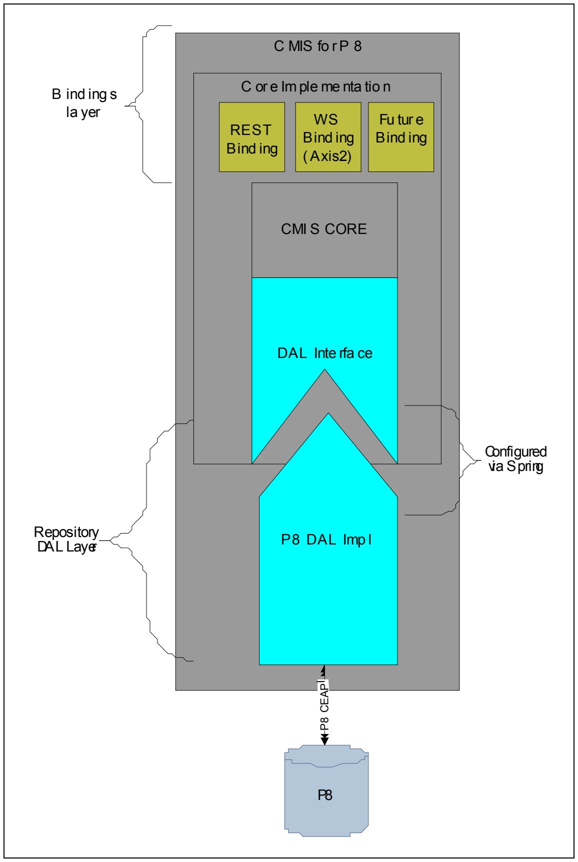


Figure 5-6 FileNet CMIS architecture

FileNet's CMIS core implementation Database Abstraction Layer (DAL) connects to the FileNet DAL, which in turn communicates with the FileNet APIs. Requests received by the Web Services and REST layers are then translated into the native P8 API by the DAL. IBM FileNet uses Java APIs to communicate with the services, and IBM FileNet uses EJB as the transport layer.

The full specification of CMIS and other useful information about the standard is at the following web site:

http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=cmis

Like all valid CMIS implementations, FileNet supports all of the required functionality defined by the CMIS specification for both the REST and SOAP bindings. There are four built-in types:

- Document: Has properties, can optionally have a content stream, and might have versions. Documents do not have to be filed in a folder and can be filed in multiple locations. Multi-filing is an optional part of the CMIS specification.
- Folder: Contains documents and folders with one and only one parent. Might contain specific typed objects.
- Relationship (optional): Has a source and a target (two objects) in a directional relationship.
- Policy (optional): Has a source and a target (two objects) in a directional relationship.

Additional subtypes might be defined. There are also properties that might have zero or more typed values. The FileNet CMIS implementation implements all of the required features defined in the OASIS CMIS specification as well as some optional ones.

5.3.2 Integration and connection points

Typically, CMIS client applications allow users to configure repository connection information, such as URL, user name, and password. After it is connected, users view folders and documents and other information provided by the repository.

CMIS allows the following services for object types and repository information:

- Create, Read, Update, Delete for all object types
- Filing in zero, one, or multiple folders
- Navigate the folder hierarchy
- Versioning operations such as check in and out, and viewing the version series
- Search, including full text search.

An interesting CMIS application developed by IBM is an easy-to-use plug-in for Mozilla Firefox. Built by IBM developers based on a Lotus solution, it is a very simple tool that integrates easily into the browser UI.

In Figure 5-7, users can browse the repository in a side panel of the browser. They can open, checkout, and delete document and view properties. FileNet's repository controls the security of the documents. The CMIS application connecting to FileNet can only access documents the authenticated user has permissions on.

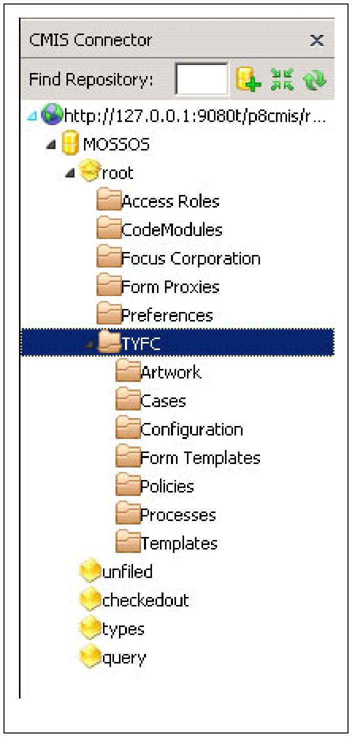
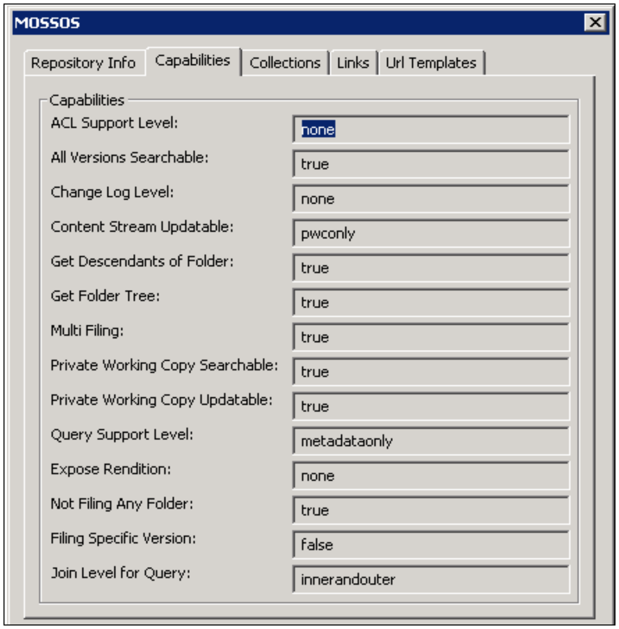


Figure 5-7 Plug in for Mozilla Firefox browsing repository

Using the CMIS protocol, the documents and their metadata are directly viewable from the browser navigation view. This application was created rapidly and required minimum product-specific configuration and coding to develop.

As in other tools, after connected, users can view information about the repository. Figure 5-8 on page 129 shows functionality exposed by the pre-release FileNet CMIS implementation.



Capabilities

ACL Support Level:	none
All Versions Searchable:	true
Change Log Level:	none
Content Stream Updatable:	pwconly
Get Descendants of Folder:	true
Get Folder Tree:	true
Multi Filing:	true
Private Working Copy Searchable:	true
Private Working Copy Updatable:	true
Query Support Level:	metadataonly
Expose Rendition:	none
Not Filing Any Folder:	true
Filing Specific Version:	false
Join Level for Query:	innerandouter

Figure 5-8 Capabilities, repository information, and other useful attributes from a pre-release version

This tool and others like it are useful for development and testing for CMIS applications and support.

5.3.3 CMIS summary

This specification was approved by the OASIS standards committee in 2010. IBM ECM implemented CMIS support in the FileNet platform and is continuing developing further products. This expands the ability of users to use their existing systems in innovative ways with better interoperability between ECM products regardless of the vendor. Organizations now have more choice and lower costs when implementing basic ECM solutions. The CMIS standard also opens new possibilities for business partners and systems integrators to easily and effectively integrate with IBM ECM tools and repositories.

5.4 Content Federation Services

The Content Federation Services (CFS) solution allows for integrating content from disparate repositories by creating references in the Content Engine’s master catalog that are used to retrieve documents stored in external repositories. With

CFS, the federated metadata appears as native objects in the Content Engine where it can be retrieved, viewed, and managed from the P8 Platform while the content remains hosted in its original repository. The federation process is managed by a set of rules and data maps that define how to map data from the source document to the target document property values in the Content Engine catalog and defines the selection criteria that determines the content that will be federated from the source repository to the Content Engine. Content can now be managed in place with consistent enterprise wide policies and control using the Content Engine. Using CFS, organizations can reduce risk and costs by avoiding forced content migrations and associated application rewrites and redeployments. A key example is the ability to deploy an enterprise level records management solution while avoiding moving existing data in other content repositories and associated applications. CFS provides a fast and effective solution to common issues related to dispersed content in an enterprise.

CFS natively supports the following configurations for federation:

- Content Federation Services for Image Services (CFS-IS)
- Content Federation Services for OnDemand (CFS-OD)
- Content Federation Services for IBM Content Integrator (CFS-ICI)

5.4.1 Architecture

Depending on the source repository, CFS might use a different approach for connectivity and configuration between the source repository and the FileNet P8 environment. As shown in Figure 5-9 on page 131, the one common object used in all configurations is a Fixed Content Device (FCD). An FCD represents an external storage device and uses the SDK provided by an independent software vendor to access the source repository. All other configurations have variations in their components and how they are configured.

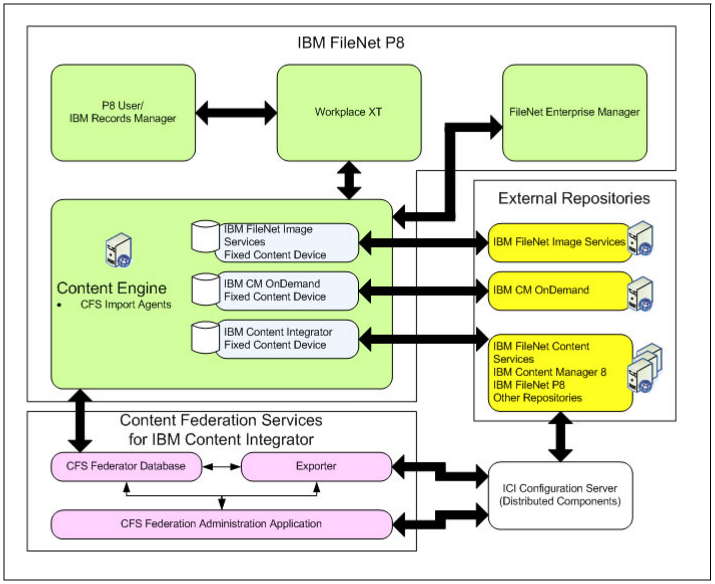

IBM FileNet P8
P8 User/
IBM Records Manager
Workplace XT
FileNet Enterprise Manager
Content Engine
CFS Import Agents
IBM FileNet Image Services Fixed Content Device
IBM CM OnDemand Fixed Content Device
IBM Content Integrator Fixed Content Device
External Repositories
IBM FileNet Image Services
IBM CM OnDemand
IBM FileNet Content Services
IBM Content Manager 8
IBM FileNet P8
Other Repositories
Content Federation Services for IBM Content Integrator
CFS Federator Database
Exporter
CFS Federation Administration Application
ICI Configuration Server (Distributed Components)

Figure 5-9 Content Federation Services conceptual architecture

Content Federation Services for IBM Content Integrator

Content Federation Services for IBM Content Integrator (CFS-ICI) uses IBM Content Integrator (ICI) connectors to provide an interface to external content providers. A connector is responsible for translating client requests into repository-specific API calls. The following connectors are supported:

- IBM FileNet Content Services
- IBM Content Manager 8
- Another FileNet P8 System
- EMC Documentum
- OpenText Livelink

CFS-ICI also supports the use of custom connectors developed using IBM Content Integrator Service Provider Interface (SPI). Developers who use the SPI to create new connectors must be familiar with the content integration API and the functional requirements needed for connector support of CFS-ICI. CFS-ICI is configured with the following components:

- Federation Administration Application: A web-based application used to configure exporter options for federation, manage federator databases, define federation rules that set the criteria for selecting external content and federating to the Content Engine master catalog, and manage rules by setting import/export schedules and monitoring rule execution processing.

- Federator Database: Database used by CFS-ICI to implement a persistent queue used by the CFS Exporter to pass import requests to the CFS Importer, store and monitor document IDs and the last updated times of federated documents used to prevent duplicate federation of source documents, and store definitions, schedules, and execution status of federation rules.
- Exporter: A stand-alone CFS-ICI application that executes federation rules against an external repository, populates the federator database with update requests that match the criteria specified in the rule, maps properties from the external repositories to P8 properties using mapping rules defined in ICI, and defines the batch size used for federation execution
- Importer: Agent that runs as part of the Content Engine that retrieves batch and content information from the federator database, creates federated content in FileNet P8, and maintains a mapping of the external version series data to the FileNet P8 version series in the federator database.

Content Federation Services for Image Services

Content Federation Services for Image Services (CFS-IS) has a tighter integration model between Image Services and the FileNetP8 Repository and does not rely on IBM Content Integrator connectors for communication between systems. The connector logic is built into the products. CFS-IS is configured with the following components:

- FileNet Enterprise Manager: Used to create a schema map to map Information Services (IS) classes and properties to FileNet P8 classes and properties
- Exporter: The IS Index Service provides real-time exporting function as documents are created, updated, or deleted in IS. The IS Remote Administration Console provides a batch mechanism to manually export or re-export documents to a FileNet P8 object store.
- Importer: The importer for CFS-IS is the CFS-IS Import Agent that runs as part of P8 Content Engine. The import agent also performs the mapping function defined in the schema map as the documents are federated into P8.
- Federation Database: Database located on the Information Services system used to store information regarding the export queue for documents and annotations.

Content Federation Services for OnDemand

Content Federation Services for OnDemand (CFS-OD) provides the ability to federate documents from IBM Content Manager OnDemand and view the federated documents in the FileNet P8 Platform. CFS-OD uses the OnDemand Web Enablement Kit (ODWEK) for communication to the OnDemand Server and does not rely on the IBM Content Integrator connectors for communication between systems. ODWEK is an add-on component that provides access to OnDemand content through a web browser or custom application. CFS-OD is configured with the following components:

- FileNet Enterprise Manager: Used to create a schema map to map OnDemand (OD) application groups and fields to FileNet P8 classes and properties.
- Exporter: A stand-alone IBM Content Manager OnDemand application that performs queries against the OnDemand database and inserts federation request in the CFS federation database. The exporter is scheduled to run using operating system scheduling utility. This occurs using cron in UNIX and the Windows Scheduler in Windows.
- Importer: The importer for CFS-OD is the CFS Import Agent that runs as part of the P8 Content Engine.
- Federator Database: Database used by the CFS-OD exporter to create federation requests and store document mapping information between the OnDemand system and the P8 repository. The CFS Import Agent processes the requests, federating the documents and deletes the entries from the database.

5.4.2 Operations

CFS offers the following operations:

- Create: Items federated from a source repository trigger the creation of a corresponding item in the FileNet P8 repository.
- Retrieve: Federated content from external repositories can be retrieved and viewed through the Content Engine using clients developed using the P8 API, such as Workplace XT.
- Update: Content Federation Services does not support the federation of mutable content. If a change is made to native content, a new version must be created and federated to FileNet P8. Mapped properties of federated documents can be updated in the source repository and the changes are reflected in the FileNet P8 master catalog.
- Delete: Allows for delete operations to be triggered from the P8 Platform, which deletes the respective document in the source repository.
- Lockdown: Used in Federated Records Management to update the security setting for a document in the source repository so that delete rights are removed from all users except the administrators.
- Move: Introduced in the 5.0 release of CFS-ICI only, the move operation allows for the content to be migrated from the source repository to a FileNet

P8 repository. After the content is successfully migrated to FileNet P8, the original document is deleted from the source repository. This function can be used for full migrations from third-party vendors to the FileNet P8 Platform.

5.4.3 Content Federation Services summary

IBM Filenet Content Federation Services provides access to content stored in various repositories across the enterprise. With CFS, federated content appears as though native to the FileNet P8 Content Engine. This provides one point of access to retrieve, view, and manage content in their external repository. The 5.0 release of Content Federation Services for IBM Content Integrator also introduces a *move* function that introduces the ability to transfer content from third-party repositories into the FileNet P8 system. Whether the content is federated or moved, CFS also enable external content to take advantage of Business Process Management or Federated Records Management capabilities of the FileNet P8 Platform.

5.5 Summary

Connector and federation expansion products for the IBM FileNet P8 Platform provide a variety of ways to incorporate content from assorted sources. The documents do not necessarily have to be moved from original locations, allowing corporations to preserve existing systems and environments. Assets can also be migrated to consolidate resources. Content can be classified, integrated, controlled, and reused in new ways, enabling businesses to make greater use of these important assets.

6

Expansion products for Information Lifecycle Governance

In this chapter, we discuss products that support and extend the IBM FileNet P8 Platform. This includes solutions to make content compliant with internal and external retention requirements, to automate its organization, to provide insight and identify patterns across unstructured data, and to make content accessible for legal and other business purposes for Information Lifecycle Governance.

We cover the following topics in the chapter:

- 6.1, “Information Lifecycle Governance overview” on page 136
- 6.2, “IBM Enterprise Records” on page 137
- 6.3, “IBM Classification Module” on page 144
- 6.4, “IBM Content Analytics” on page 151
- 6.5, “eDiscovery Manager and eDiscovery Analyzer” on page 157
- 6.6, “Summary” on page 163

For an overview of all of the IBM FileNet P8 expansion products, refer to 4.1, “Expansion product overview” on page 92.

6.1 Information Lifecycle Governance overview

Businesses must be able to organize, manage, understand, and find the information housed in their systems to effectively support compliance, electronic discovery, and content-driven business activities. In other words, businesses must be able to govern the entire life cycle of information, which is a field known as *Information Lifecycle Governance*. The expansion products for Information Lifecycle Governance refer to the integrated software for content assessment, collection and archiving, classification, records management, and eDiscovery:

- IBM Content Collector
- IBM Enterprise Records
- IBM Classification Module
- IBM Content Analytics
- IBM eDiscovery Manager and Analyzer

IBM Content Collector is a key component of Information Lifecycle Governance solutions, enabling the automated collection of information from file shares, email systems, and SharePoint sites in support of compliance. For details, refer to 4.3, “IBM Content Collector” on page 94.

Some of these products take advantage of and extend the core platform as P8 application modules, while others can operate externally to P8. In both cases, the additional capabilities integrate seamlessly, enabling organizations to derive further value out of their P8 investment by exploiting their existing content infrastructure in additional ways. Each of these solutions are described in the sections in this chapter.

6.1.1 Compliance

Compliance describes how well an organization enforces and adheres to policies and procedures, whether internally defined or externally mandated by industry, financial, or government standards. Compliance management refers to the technologies applied to implement processes for reducing risk, enabling quicker response to legal inquiries, establishing trust, and leveraging information in a manner that can be proven to satisfy established guidelines. IBM Information Lifecycle Governance offerings include capabilities to manage documents and other content throughout their lifespan, regardless of type, format, media or storage location. When implemented as part of an enterprise compliance strategy, these solutions drive more efficient electronic discovery, reduce legal exposure from inconsistent and error-prone manual processes, and can significantly reduce storage and discovery costs associated with retaining obsolete and irrelevant content.

As defined by the Federal Records Act, a record is:

"...recorded information, regardless of medium or characteristics, made or received by an organization that is evidence of its operations and has value requiring its retention for a specific period of time.[1]"

A record can be either electronic or physical as long as it has intrinsic or regulatory relevance to the organization that warrants its retention. Examples of electronic records include digital documents and files, emails, customer data, and even customer or organizational communication conducted on social media sites. Physical records can include books, photographs, tapes or DVDs, and any hard copy document. Although there are a few industries in which it is not the case, in the majority of organizations, not all documents and files are treated as records.

Records management ensures that important records remain available over appropriate periods of time for reference by internal users or external auditors. It is a formal and structured process of identifying record information, preserving needed content, and destroying content that is no longer needed. Destruction of a record is allowed only after the approved retention period is reached. To be valid, the system of record must provide the security and auditing controls to prove the reliability, authenticity, integrity, and usability of records.

6.2 IBM Enterprise Records

Formerly named IBM FileNet Records Manager, IBM Enterprise Records is an optional IBM FileNet P8 product that manages document life cycles according to retention rules. IBM Enterprise Records provides records governance to facilitate trustworthiness and accuracy, and helps ensure compliance with defined records retention policies. Enterprise Records supports efforts to reduce regulatory compliance, business operation, and discovery risks by subjecting all records-class content to centralized preservation, hold order, and life-cycle disposition policies. Key capabilities include:

- File plan administration: Central and departmental file plan administration for both electronic and physical records from a single administrative module.
- Federation: Administers content in a federated environment to manage records in place in both IBM and non-IBM repositories with no migration of records required.
- Support for multiple file plans: Allows organizations to apply separate records rules based on geography, business line, or other requirements demanding entirely separate file plans.

[1] http://www2.ed.gov/policy/gen/leg/fra.html

- Reporting: Administrative reporting against the most common records activities, including declarations, and holds.
- Holds: Provides a single interface to create, manage, apply, and audit holds, including conditional or *dynamic holds*, to suspend the disposition of records that might need to be preserved as part of a pending investigation.
- Consistency: Invisibly enforces consistent compliance and records management policies throughout an enterprise, offloading the burden from end-users of deciding what constitutes a record.
- Standards-based: Certified against the U.S. DoD 5015.02 Standard for Records Management and supports ISO 15489, and VERS.
- Enterprise-ready: Operates as an extension of P8 Content Manager and integrates directly with IBM Classification Module and Content Collector.

6.2.1 Architecture

Enterprise Records builds on the IBM FileNet P8 Platform by leveraging and extending the services provided by the core engines described in earlier chapters. Figure 6-1 on page 139 shows several of the major Enterprise Records components within the IBM FileNet P8 architecture and their relationship to the underlying platform services.

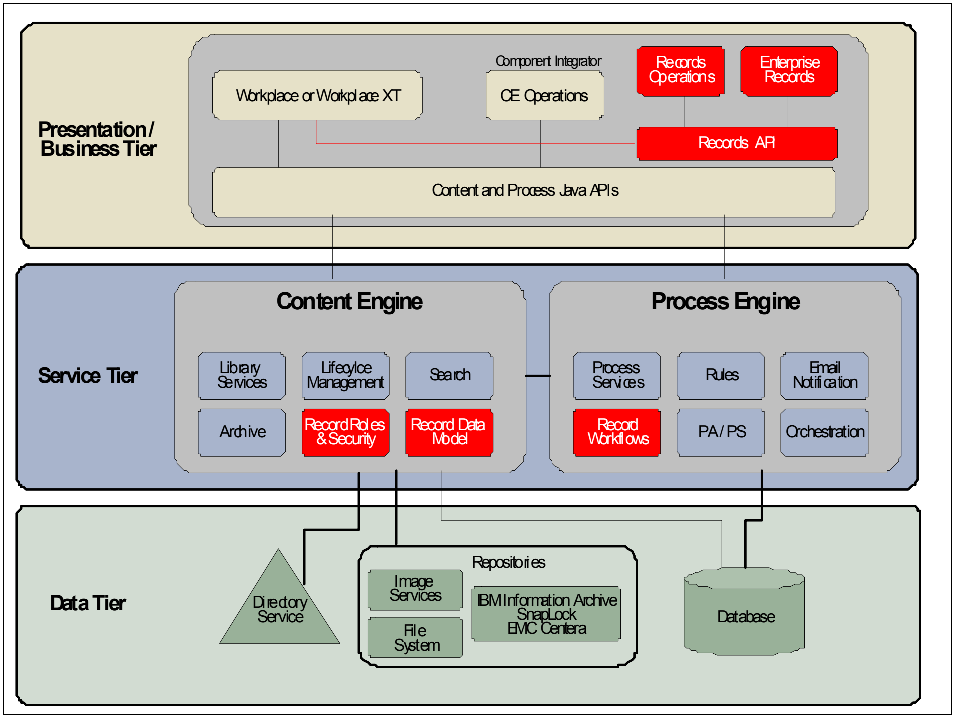


Figure 6-1 High level Enterprise Records architecture

The core Enterprise Records components in Figure 6-1 are:

- In the service tier, within Content Engine:
  - Enterprise Records data model: This component provides the core definitions for Enterprise Records business objects, such as record classes, records containers, and disposition schedules, upon which the records management system is built.
  - Enterprise Records roles and Content Engine security: The records management security capabilities are built upon the underlying Content Engine security model coupled with default Enterprise Records security roles that determine functional user access. Enterprise Records takes advantage of Content Engine marking sets to implement certain features of its security model.
- In the service tier, within Process Engine:
  - Enterprise Records workflows: Process Engine provides special records management-related workflows for implementing a variety of disposition actions. Workflows take advantage of the full capability of Process Engine and can be completely customized.

- In the presentation/business tier:
  - Enterprise Records Web application: This component provides core administrative and management functions for the file plan and the records that it contains.
  - IBM FileNet Workplace/Workplace XT Web application: Workplace and Workplace XT are integrated with Enterprise Records for automated and manual record declaration.
  - Records API (also known as Records Manager Java API): This API exposes the Enterprise Records functions for custom application development.
  - Component integrator (Enterprise Records operations): This component integrates records management functionality into a BPM environment and thus records-enables business processes.

Application Engine and Workplace naming convention: Application Engine is the official name for Workplace. Application Engine does not equate to Workplace XT. Both Workplace and Workplace XT support a common set of functions but differ in other areas. For consistency of the terminology used in the book, we use Workplace instead of Application Engine throughout the book.

Figure 6-2 on page 141 illustrates Enterprise Records integrated with the IBM FileNet P8 Platform, the relationship between records (which store the record-related metadata of the declared documents) in the File Plan Object Store (FPOS), and the associated declared documents in the Records-enabled Object Store (ROS).

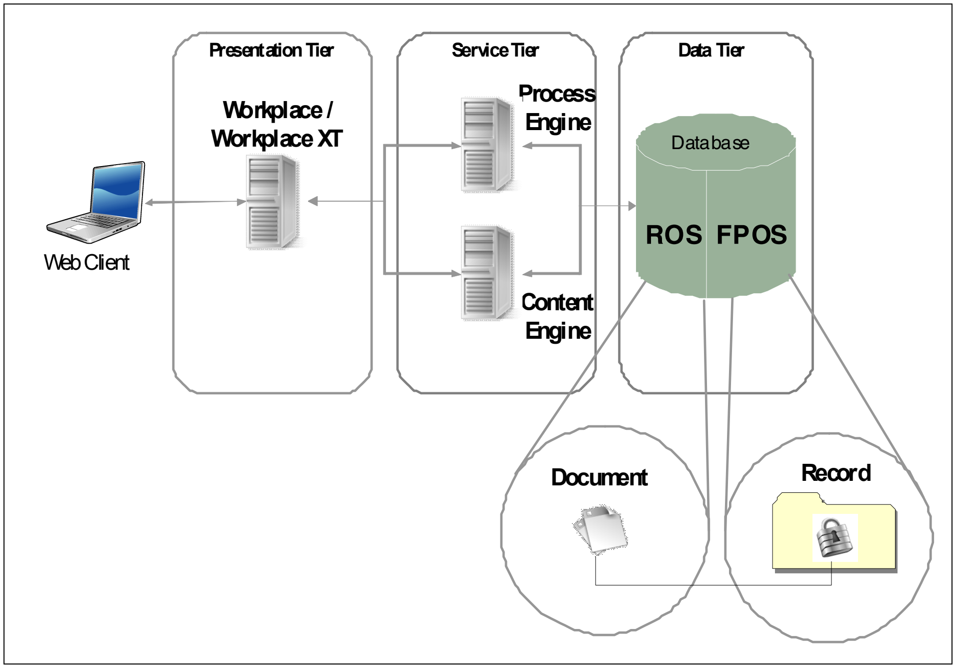


Figure 6-2 Integrated file plan and content repository architecture

It is important to note that a record is an entirely separate object from the associated declared document. The record must be contained in a file plan within the File Plan Object Store (FPOS). The record object has a direct reference to the actual declared document that exists in the Record Object Store (or a reference to a physical artifact that exists outside of the ROS). The record object acts as a security proxy for the declared document that it references. For documents that reside in an IBM FileNet P8 content repository, the repository is simply another object store that is records-enabled and is managed by Content Engine. Other electronic documents can reside in a repository outside of IBM FileNet P8 content repository, and are managed by a federated connection. In the case of physical artifacts, the records in Enterprise Records serve as markers or references to the physical objects with relevant metadata to identify and manage them as records.

6.2.2 Federated Records Management

There can be situations where records are generated and stored in a system other than an IBM FileNet P8 content repository. A number of common content stores—including file systems, email systems, and some ECM repositories—might lack the security, immutability, and other controls required for the long term storage of records that other repositories do. In these cases, Enterprise Records can be configured to communicate with these systems such that records can be identified and managed in place by the Enterprise Records file plan.

Leveraging P8 Content Manager's Content Federation Services, Enterprise Records can manage documents and content that do not reside in a P8 Content Manager repository. As depicted in Figure 6-3, Records Federation is accomplished with federation index that maintains an index of the objects found outside of P8. This means that a file plan administered in Enterprise Records can manage records in other IBM FileNet P8 repositories and in IBM Content Manager, IBM Content Manager on Demand, and IBM FileNet Image Services. The result is far simpler records administration, as separate records systems do not need to be configured and maintained. Federated record capabilities include declarations, holds, retention schedules and dispositions.

Figure 6-3 shows the records federation model.

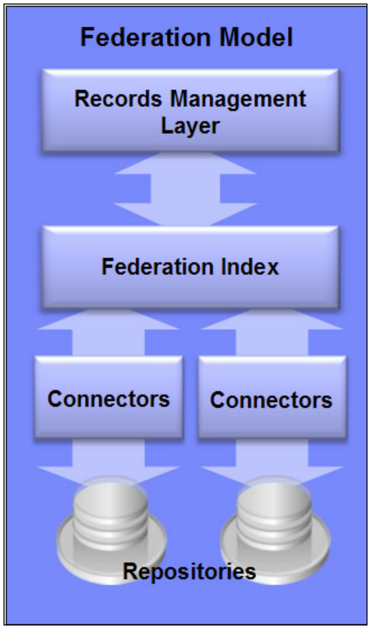


Figure 6-3 Records federation model

Pre-integrated federation connections also exist for EMC Documentum and OpenText. The IBM Content Integrator can also be used to build federated records connections to more than 30 additional repositories.

Systems lacking the necessary safeguards, security, and controls to serve as a system of record must not be federated and can instead be candidates for migration into a P8 Content Manager repository. Chapter 4, “Expansion products for content ingestion” on page 91 described the Content Collector family of products for exactly this purpose, allowing unsecured content to be moved under the control of P8 for long term content and records management.

6.2.3 Application Programming Interface

Enterprise Records provides an API known as the Records Manager Java API. The API provides Java-based access to commonly used objects and includes methods for performing record-related operations, such as record declaration, file plan navigation, and record disposition. The API can be used to customize an application that requires records management functionality or to develop new functionality to expand on the features that already exist.

The Records Manager Java API leverages the Content Engine Java API. The Content Engine high-level objects provide access to object store content persistent in a Content Engine. When working with the Records Manager Java APIs, the basic objects include the Content Engine high-level objects and the Records Manager Java API base objects.

6.2.4 Bulk records operations

Bulk Declaration Service (BDS) is a mechanism to perform multiple record operations in batches. BDS is implemented using a set of interfaces and classes known as the BDS API. The following BDS bulk operations are available:

- Declaration of electronic records from existing documents in Content Engine
- Declaration of physical records
- Creation of new Content Engine documents and declaring them as records
- Creation of new Content Engine documents

Most BDS operations can be performed using Enterprise Records APIs, too, but BDS tends to offer superior performance.

The power of BDS lies in batch processing, which results in a savings of overall execution request time. BDS is extremely useful when a new records management environment is being set up and large numbers of documents must be declared as records. A custom, stand-alone application can be developed using the BDS API to declare records quickly.

For BDS interfaces and classes detail, refer to ecm_help.

6.2.5 Dependencies

Reporting is an optional feature. IBM Enterprise Records can use Crystal Reports (which must be purchased separately) to support reporting capabilities.

For a complete list of system requirements, refer to the IBM FileNet Hardware and Software Requirements documentation at:

http://www.ibm.com/support/docview.wss?rs=3278&uid=swg27013654&S_CMP=rnav

6.2.6 Enterprise Records summary

By adding Enterprise Records to their P8 environment, organizations can introduce robust records management for their unstructured content, extracting greater value out of their P8 infrastructure. Enterprise Records combines content, process, federation, and connectivity to streamline both electronic and physical records operations, regardless of where records reside.

6.3 IBM Classification Module

Given the volumes of email, documents, and countless other forms of electronic content within an organization, proper metadata tagging and classification is critical to ensure that information can be reliably located, used, and preserved. Formerly called Classification Module, IBM Classification Module automates the organization of unstructured content by analyzing the full text of documents and emails. Classification Module can accurately and automatically classify information to make it more accessible and to help reduce the risks and costs of manual filing.

IBM Classification Module can organize information in P8 Content Engine by policies or key words and assign metadata that is based on the full context of the document. The classification process does not just search for a single word or phrase, but analyzes the entire document, distills the main concept of the text, and assigns the text to a category. When analyzing content, IBM Classification Module can recognize and compensate for misspellings, abbreviations, jargon, and technical terms.

Accuracy improves over time because the system adapts to the unique nature of the content of each business by identifying categories from examples it is provided. When feedback is received, the system adjusts in real time and immediately implements any corrections that are made, meaning that the accuracy of the classification results keeps pace with changes in the business.

Classification Module uses natural language processing to analyze the content of documents and emails to categorize them. It learns to categorize from examples in an enterprise or from keywords and applies rule-based and context-based classification using text analysis with rules. Classifying documents and other content as it is ingested into the P8 Content Engine allows it to be properly filed for search and retrieval. With direct integration to Content Collector, IBM FileNet P8 and Enterprise Records, Classification Module can automate the identification, capture, classification and retention or documents.

Benefits of IBM Classification Manager include:

- Unburdens end-users and ensures consistency by reducing manual categorization tasks
- Prevents emails and documents with limited business value from clogging archives
- Organizes content with consistent, reliable and auditable logic
- Adapts to changes in policies and categories by incorporating user feedback in real-time
- Automates classifications with high accuracy by combining multiple methods of classification, ranging from keyword rules and proximity matching to pattern extraction to highly accurate 'learn-by-example' context-based approaches
- Natural language processing support for 18 languages, including English, French, German, Italian, Spanish, Dutch, Swedish, Portuguese, Russian and Simple Chinese

In addition to providing automated classification services for IBM FileNet P8, Classification Module also supports the classification of documents and emails in file systems and IBM Content Manager repositories.

6.3.1 Integration and connection points

Classification Module is integrated into IBM FileNet P8 for bulk classification of content upon ingestion or reclassification of content that is already under management. The Classification Center allows rules to be created that specify the action that must be taken on documents after they are classified. As shown in Figure 6-4 on page 146, this includes decisions specific to how the documents must be handled by P8, including the assignment of the appropriate document class, filing them into folders, and declaring them as records (if using Enterprise Records).

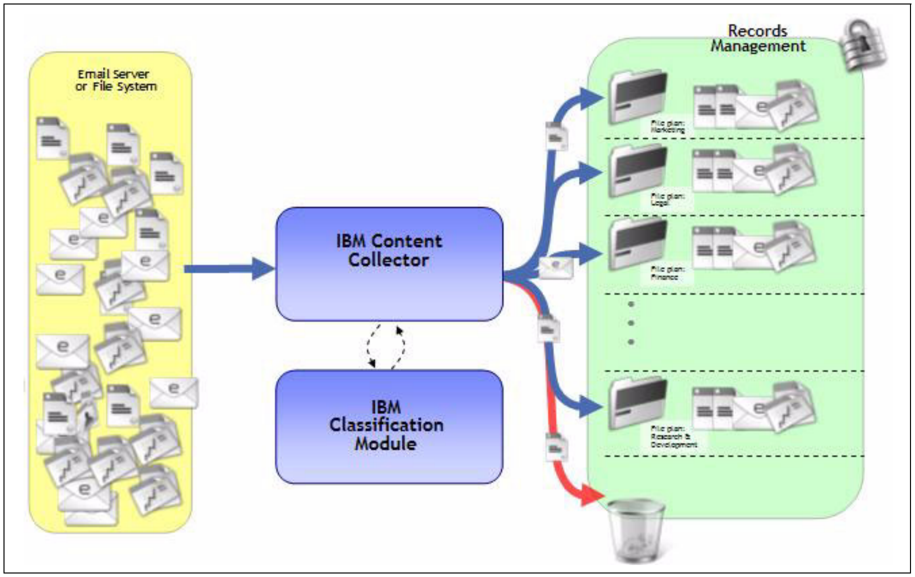


Figure 6-4 Classification Module and Content Collector automate records classification

Classification Module also integrates directly with Content Collector, providing classification or additional metadata, which can then be used by Content Collector to determine whether an item is captured and how the item is processed if it is captured and filed. With Classification Module, advanced classification can be applied to the content ingestion process from SharePoint, file systems or email systems rather than relying solely on document metadata or source location metadata, for example, Classification Module can distinguish important emails from those that have no business value. Based on the content analysis that Classification Module performs, Content Collector can then determine the appropriate action to be taken, including the removal, capture, or declaration as a record of the document or message.

IBM Classification Module runs a set of server-side processes on one or more servers and provides a number of client libraries for remote access that are designed for various development environments.

Organizations can use the client libraries that best fit their application development environment. Client libraries are available for C/C++, Java, Visual Basic (ASP) and .NET (WSDL file for .NET connectivity) development, each of which is documented with the product. All of the client libraries are based on SOAP. All libraries define the same basic set of structures (objects) and functions (methods).

The API documentation provides a complete list of API objects, methods, and their parameters. Help is available in several formats for the development of C, Java, and COM clients. For developing applications with .NET connectivity, consult the COM API documentation. Samples that demonstrate system functionality and how to use the various client libraries are provided with the product.

6.3.2 Architecture

The IBM Classification Module is a distributed, scalable platform for providing classification services to support content management, with a three-tier architecture that consists of data, application, and presentation layers. Figure 6-5 depicts the components and their respective interactions within a Classification Module implementation.

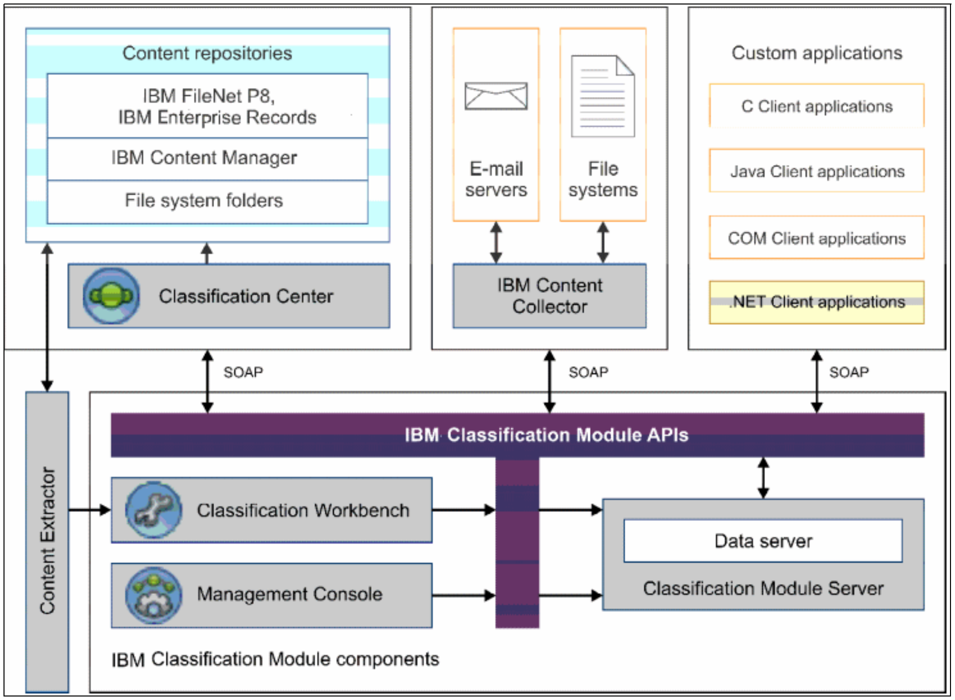


Figure 6-5 Classification Module architecture

The data layer stores configuration information. The application layer, which is the core of the system, provides various services, such as language analysis, instance pooling of knowledge bases, and management of the distribution of tasks across computers. The presentation layer communicates with the application layer through a client API and performs data layout and formatting actions. In addition, applications can interact with the system by using the SOAP protocol. Figure 6-6 shows the data flow of the Classification Module learning process.

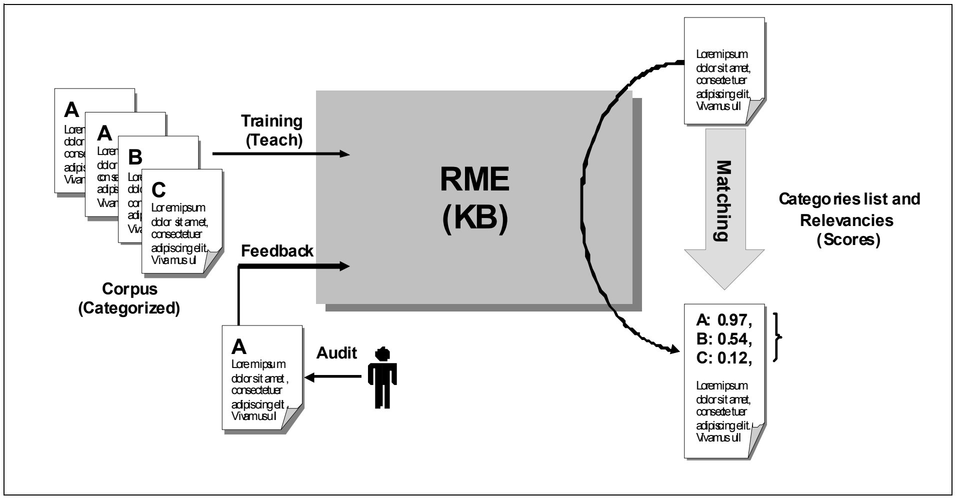

A
A
B
C
Loremipsum dolor sit amet, consectetuer adipiscing elit, Vivamus ul
Corpus (Categorized)
Training (Teach)
Feedback
RME (KB)
A
Loremipsum dolor sit amet, consectetuer adipiscing elit, Vivamus ul
Audit
Loremipsum dolor sit amet, consectetuer adipiscing elit, Vivamus ull
Matching
Categories list and Relevancies (Scores)
A: 0.97,
B: 0.54,
C: 0.12,
Loremipsum dolor sit amet, consectetuer adipiscing elit, Vivamus ull

Figure 6-6 Classification Module learning data flow

To teach the knowledge base how to categorize the documents, a training set is loaded into the Classification Workbench, which is the application that is used to create and train a knowledge base. Feedback given to Classification Module changes the way text and properties are evaluated and processed in the future, which changes the category list and relevancy of the properties, resulting in better future matching.

Figure 6-7 shows the Classification Module server architecture.

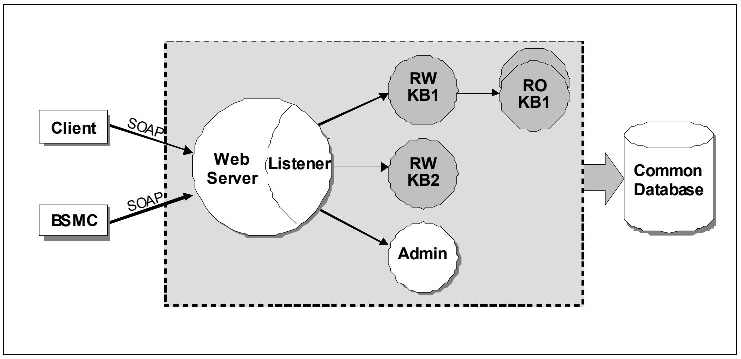


Figure 6-7 Classification Module server architecture

Supported Web servers are either IIS or Apache. The listener that is inside the Web server is responsible for routing requests to the appropriate components. RW KB 1 and RW KB2 in the Adobe® figure stand for Knowledge Base Read/Write 1 and 2, and they are responsible for read and write requests on a specific knowledge base. RO KB1 is responsible for read only requests on a specific knowledge base. The administration process is responsible for server administration requests, and the Common Database is the persistent data store for configuration information, history, and other data.

6.3.3 Components

IBM Classification Module consists of various knowledge bases, rules, and graphical interface tools to help organizations set up and administer an automated classification system, including tools that enable the direct integration of classification activities between Classification Module and an IBM FileNet P8 or IBM Content Manager system.

Knowledge base

IBM Classification Module works with an adaptive knowledge base, which is a set of collected data that is used to analyze and categorize content items. The knowledge base reflects the kinds of data that the system is expected to handle.

Before the knowledge base can analyze text, it must be trained with a sufficient number of sample content items that are properly classified into categories. A trained knowledge base can compute a numerical measure of an item's relevancy to each category. This process is called matching or categorization. The numerical measure is called relevancy or score.

The performance of a knowledge base can be maintained and improved over time by providing it with feedback (confirmation or correction of the current categorization). The feedback is used to automatically update and improve the accuracy of the knowledge base over time. This process of automatic self-adjustment is called learning. Learning is an incremental process, where recently learned information has a greater impact on classification.

Decision plan

A decision plan is a collection of rules configured to determine how IBM Classification Module classifies content items. Unlike a knowledge base, which is built automatically by supplying a sample set of categorized content items, a decision plan is built by configuring one or more rules.

Rules consist of triggers and actions. A trigger determines the conditions that must be met to initiate an action. An action determines how the document is to be classified.

Field definitions

As part of the process of creating a knowledge base, the administrator specifies the fields in the content set that contain data that is to be analyzed or evaluated when the content is classified. To support flexible text input, the system must identify the content types and data types of the text that it receives. IBM Classification Module receives text as a series of fields. A field definition defines the data type and content type of each field. A common collection of fields is shared by all knowledge bases in an IBM Classification Module system.

6.3.4 Classification tools

IBM Classification Module provides several graphical interfaces to help administer the system, create and train knowledge bases, and classify content in IBM FileNet P8 repositories.

Management Console

The Management Console is the application used to administer a Classification Module system. For example, knowledge bases and decision plans can be added to the Classification Module server, field definitions added and modified, and information viewed about the servers that host Classification Module components.

Classification Workbench

Classification Workbench is a Windows application for creating knowledge bases and decision plans, analyzing knowledge bases and decision plans, and evaluating the performance of knowledge bases and decision plans through reports and graphical diagnostics.

Taxonomy Proposer

The Taxonomy Proposer is installed with Classification Workbench and is a Windows application used to discover new categories in a body of documents. This tool is useful in situations where there is no existing taxonomy or where suggestions for how to categorize a collection of documents are needed.

Classification Center

The Classification Center is used to manage the classification of content that is stored in IBM FileNet P8. This Web application is used to select the content to be classified, configure classification options (such as the decision plan to use and various runtime preferences), monitor classification activity, and view the classification results. The Classification Center can also be used to reclassify documents.

Content Extractor

Content Extractor is a command-line tool to extract the content from an IBM FileNet P8 object store or IBM Content Manager repository. The extracted content can be imported into Classification Workbench and used to train a knowledge base or provide test data for a decision plan.

Client applications

Client applications for IBM Classification Module interact with knowledge bases and decision plans in a variety of ways, for example, applications can:

- Create and configure knowledge bases and decision plans
- Match texts against a knowledge base
- Retrieve decision results from a decision plan
- Submit feedback to a knowledge base or decision plan
- Identify the language of texts
- Support multilingual applications

6.3.5 Classification Module summary

IBM Classification Module is a platform for automating decision making in the enterprise content management architecture. Organizations need a standard and consistent way to understand and access the unstructured information that resides in their various file systems and content management repositories. IBM Classification Module makes useful content searchable and able to be found quickly, resulting in faster business decisions, while helping to ensure compliance with content retention policies.

6.4 IBM Content Analytics

Formerly named Cognos Content Analytics, IBM Content Analytics helps companies discover, refine, visualize, and deliver new business insights through the analysis of unstructured content, including data stored in P8 Content Engine. Content Analytics gives organizations the necessary tools to access and analyze the volumes of unstructured content that are locked up in documents, notes, emails, Web pages, and more.

IBM Content Analytics allows risk professionals, market researchers, customer service organizations and other strategic information consumers to access virtually any type of structured, semi-structured or unstructured content found within the enterprise. Using Content Analytics, quality issues can be identified earlier, competitive insights learned sooner, and trends detected that might fall outside of the normal business reporting channels. Content Analytics enables business users to discover, refine and deliver new insights by highlighting patterns, anomalies, and deviations through advanced visualizations and by analyzing unstructured content alongside standard metrics and data.

Content Analytics automatically identifies and tags key attributes and entities within unstructured content by crawling almost any content source and identifying key words and phrases. This initial analysis can then be refined with visual navigation aids and drill-down capabilities based on identified key attributes, entities, and extracted dimensions.

Example use cases where content analytics can help address a variety of information challenges include:

- Improve customer satisfaction through high-volume analysis of customer satisfaction comments and feedback
- Gain better visibility into the marketplace through automated news, survey, and brand analysis
- Better anticipate customer needs by identifying trends in unstructured customer communications
- Optimize document-intensive processes through intelligent classification and routing of content items
- Get ahead of product quality problems through complaint, warranty, repair, and support ticket analysis
- Reduce fraud by intelligently parsing forms, documents, and communications to seek out patterns indicative of criminal intent
- Enhance research and investigations through combined data and content analytics

6.4.1 Architecture

IBM Content Analytics provides crawlers for a wide variety of sources that allow content, metadata, structured data, and unstructured data to be collected and fed into the analytics pipeline. Each item is converted to text and sent through a series of processing and analysis steps. Each step annotates the content item with additional information and cleans, clarifies, and extracts meaning from the item. Figure 6-8 on page 153 depicts the flow of data throughout the analysis process.

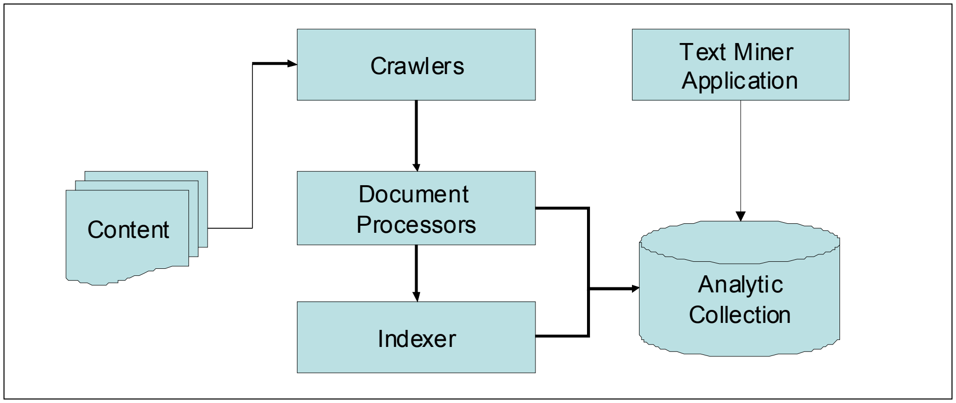


Figure 6-8 IBM Content Analytics overview

As shown in Figure 6-8, content is acquired through crawlers. The content is then processed by Document Processors. Indexer analyzes and indexes the documents to create the Text Analytics collection. Users can then search, explore, mine, and discover data using the Text Miner application.

Annotators contain the logic that analyzes a document and discovers and records descriptive data about it. IBM Content Analytics includes a powerful set of annotators to analyze and extract meaning from content. These annotators:

- Detect source language and character encoding of the content
- Tokenize the text into words
- Identify the parts of speech of words
- Find meaningful word phrases
- Extract named entities from the text automatically, such as people, locations, and organizations
- Automatically categorize and classify the content item through IBM Classification Module
- Find custom patterns in the text through customer-defined regular expressions
- Search for relevant, custom-defined dictionary terms, such as product names or brands

Because IBM Content Analytics implements the open Unstructured Information Management Architecture (UIMA) framework, the processing of content can be fully customized to take advantage of existing, both open source and commercial, UIMA annotators, enhancing the automated content analysis capabilities.

Collections

A content analytics collection represents the set of sources that users can explore and mine with a single query. When a collection is created, the sources to include are specified and options configured for how users can search and analyze the indexed data.

Multiple collections can be created, and each collection can contain data from a variety of data sources. The type of collection that is created determines which functions are available for configuring the collection.

Search collection

Search collections support search and retrieval functions, including the ability to browse and narrow results by selecting facets, sort documents by relevance or date, preview documents in the search results, and view thumbnail images of certain types of documents.

Text analytics collection

Text analytics collections support search and text mining functions, such as the ability to explore correlations, deviations, and trends in data. Data can be exported for analysis in data warehouse or business intelligence applications.

Creating and administering a collection involves colecting, analyzing, indexing, exploring, and mining data.

Collecting data

The crawler components collect documents from data sources, either on a continual basis or according to a specified schedule. Frequent crawling ensures that users always have access to the latest information.

Analyzing data

The analytics pipeline extracts text from documents, performs linguistic analysis, finds meaningful word phrases, extracts entities, and performs custom analysis on each document that a crawler crawls. The detailed content analysis provides facets of data that can be used for exploring and mining the content.

Indexing data

The index components add data from new and changed documents to the index. The index components also perform global analysis of the documents in a collection to determine correlation scoring.

Exploring and mining data

The text mining components provide an interactive graphical interface for exploring the data that was extracted from content items. Visualizations generate views that link highly correlated terms and provide real-time exploration of vast data sets.

Crawlers collect documents from data sources so that the documents can be analyzed, indexed, searched, and mined. A single collection can have multiple crawlers, and each crawler is designed to gather data from a particular type of data source.

6.4.2 Integration and connection points

IBM Content Analytics can collect and extract content from a variety of data sources.

Predefined support is available for the following data source types:

- IBM Content Manager item types (documents, resources, and items)
- IBM DB2 databases
- IBM Domino Document Manager (formerly Domino.Doc) databases
- IBM FileNet P8 object stores
- IBM Lotus Notes databases
- IBM Lotus QuickPlace® databases
- IBM Lotus Quickr content libraries
- IBM Case Manager
- Lotus Web Content Management sites
- IBM Content Integrator repositories, including Documentum, Content Services, Hummingbird Document Management (DM), OpenText Livelink Enterprise Server, and WebSphere Portal Document Manager (PDM)
- IBM WebSphere Information Integrator nickname tables for many database system types, including IBM DB2 for z/OS®, IBM Informix®, Microsoft SQL Server, Oracle, and Sybase
- IBM WebSphere Portal sites
- Microsoft Exchange Server public folders
- Microsoft SharePoint repositories
- Microsoft SQL Server databases
- Microsoft Windows file systems
- Network news transfer protocol (NNTP) news groups
- Oracle databases

- UNIX file systems
- Web sites on the Internet or within an intranet

For the latest information about supported data source types and the supported product versions, see the Supported data sources page on the product web site.

IBM Content Analytics supports IBM Cognos BI reports and models. Cognos reports can link to and from the Content Analytics Text Miner interface.

In addition to the supported integrations previously listed, IBM Content Analytics provides several options for customization that include using the LanguageWare Resource Workbench for altering the behavior of text analytics and a full-featured REST application programming interfaces.

LanguageWare Resource Workbench

LanguageWare Resource Workbench is a development tool for building UIMA-compliant text annotators. Using the Workbench, organizations can develop and deploy custom text analytics that extend or modify the way that IBM Content Analytics processes documents. LanguageWare Resource Workbench can also be used to expand on the 11 languages supported by Content Analytics by developing a compatible lexical analysis. IBM Content Analytics and LanguageWare Resource Workbench are integrated through the Content Analytics REST API.

REST APIs

IBM Content Analytics includes a set of REST APIs for invoking content analysis functions from within custom applications.

The Search REST API extends applications that require the ability to query collections and to query and browse facets. The Search REST API also includes a cube API for two-dimensional mapping. Using the Search REST API, developers can create custom applications that lists collections and facets, fetches content, and even provides thumbnails and previews of documents. The Search REST API also performs spelling corrections, synonym expansions, and can offer type-ahead suggestions to users.

The Administration REST API is intended for administrative applications where programmatic control over the Content Analytics system is needed. It supports the management of collections, the administration and monitoring of Content Analytics components, and enables the adding of documents to a collection. The Administration REST API provides developers with an alternative to using the standard Content Analytics administration utilities by incorporating their capabilities into custom applications, for example, using the Administration REST

API, an application can create or delete documents and collections on the fly, and start and stop the Content Analytics crawler, indexer, and search runtime.

6.4.3 Dependencies

IBM Content Analytics supports AIX, Red Hat and SUSE Linux, and Windows operating systems. Either a Jetty or IBM WebSphere Application Server is required. Content Analytics includes the 32-bit and 64-bit versions of Java Virtual Machine (JVM) 1.6. See the following web site for the full list of supported platforms and system requirements:

http://www.ibm.com/support/docview.wss?rs=4173&uid=swg27015092&S_CMP=rnav

6.4.4 Summary

Content Analytics supports semantic search and navigation to drill down and explore structured and unstructured data. It enables business users to discover, refine, and deliver new insights by highlighting trends and finding complex or hidden problems.

IBM Content Analytics provides organizations with the necessary tools to discover the business value contained within unstructured content. Working through a dynamic, highly visual interface, this solution discovers important information by identifying and surfacing correlations, enabling organizations to discover hidden trends, augmenting Business Intelligence reports, or enhancing business processes with unstructured content.

6.5 eDiscovery Manager and eDiscovery Analyzer

eDiscovery—the discovery of electronic information—refers to the process of locating, securing, and preparing electronic data for the purpose of using it as evidence in a civil or criminal legal case. IBM eDiscovery is an integrated solution that helps organizations locate stored information to improve litigation response across the enterprise. Leveraging the scalable P8 Enterprise Content Management platform and IBM Enterprise Records, IBM eDiscovery enables companies to manage the entire eDiscovery process in a secure, audited, and defensible manner. IBM eDiscovery tools reduce risk and cost by allowing much of the discovery process to be conducted in-house using internal Legal staff.

IBM eDiscovery consists of 2 components: IBM eDiscovery Manager and IBM eDiscovery Analyzer. Each component is explored in this chapter.

6.5.1 eDiscovery Manager

IBM eDiscovery Manager assists with legal discovery and internal investigations of possible violations of corporate policies by enabling users to locate, export, and preserve documents that might become evidence. eDiscovery Manager provides an automated hold and preservation process, first pass review, process automation, and a complete chain of custody to collect and preserve key electronic evidence. Litigation holds that prevent the destruction or modification of relevant data can be managed on a case-by-case basis.

eDiscovery Manager can search, select, and organize content for early case insight and reduce content volume for further legal review. It offers secure, audited collection and management of email and other electronically stored information. Unlike other approaches that result in the duplicate storage of documents, eDiscovery Manager allows content to remain in place in the repository when added to a case or case folder. It does so by creating a reference to the relevant documents only, and does not require that they be moved or copied.

Additionally, eDiscovery Manager provides proven auto-classification and integrates with robust records management to help IT departments manage information proactively for compliance and electronic discovery requests. It provides immediate searching, selecting, holding, and export tools that help compliance investigators and in-house counsel review, prioritize, and filter collected case materials to optimize case preparation and reduce the volume and cost of litigation review.

eDiscovery Manager provides key functions for IT response to eDiscovery:

- Creating cases and assigning users to review and manage them
- Managing electronically stored information in place for multiple cases
- Searching and culling case relevant information
- Holding and locking down the result set
- Previewing for relevancy
- Built-in change audit tracking, reporting authenticity and chain of custody
- Exporting result set for detailed attorney review

Using the eDiscovery Manager Web client, authorized IT personnel can find the documents that are relevant to a case. To find email documents, searches can be performed based on date ranges, senders, recipients, subject lines, bodies and attachments, or any combination of these. When searching the body and attachments, the search can be narrowed by creating complex search statements with keywords, phrases, Boolean operators, and wildcard characters. To find other kinds of documents, creation dates, modification dates, file names, file paths, or any combination of these can be searched.

Cases are used to organize search results in a way that is meaningful to Legal staff. Upon being assigned to a case, a legal hold is applied to the search results in a case folder. The search terms that produce the results are stored in the case so that they can be referenced or reused in the future. These are called case searches.

After search results are saved to a case folder, a legal hold is applied to all content returned. Documents can then be moved and copied between folders and declared as records if using IBM Enterprise Records. All or a subset of the search results can be exported, and these exported documents provided to a third party or opposing counsel. By default, documents are exported in their native formats. Documents can also be exported according to the Electronic Discovery Reference Model (EDRM) as Extensible Markup Language (XML) files. The EDRM XML format is an industry standard for transferring files between e-discovery applications from various vendors. The eDiscovery Manager API further supports the development of custom export plug-ins.

IBM eDiscovery Manager can search unstructured content and retrieve documents that are stored in either IBM FileNet P8 or IBM Content Manager.

6.5.2 Architecture

An eDiscovery Manager system consists of the following main components:

- Web browser
- Email client (optional, as HTML preview is available for all document types, including email documents)
- WebSphere Application Server
- Content management server
- Content Collector to archive emails, files, and SharePoint content (optional)
- Records management software (optional)

Figure 6-9 on page 160 depicts the high-level view of how eDiscovery Manager (and eDiscovery Analyzer) interacts with the other components of the architecture.

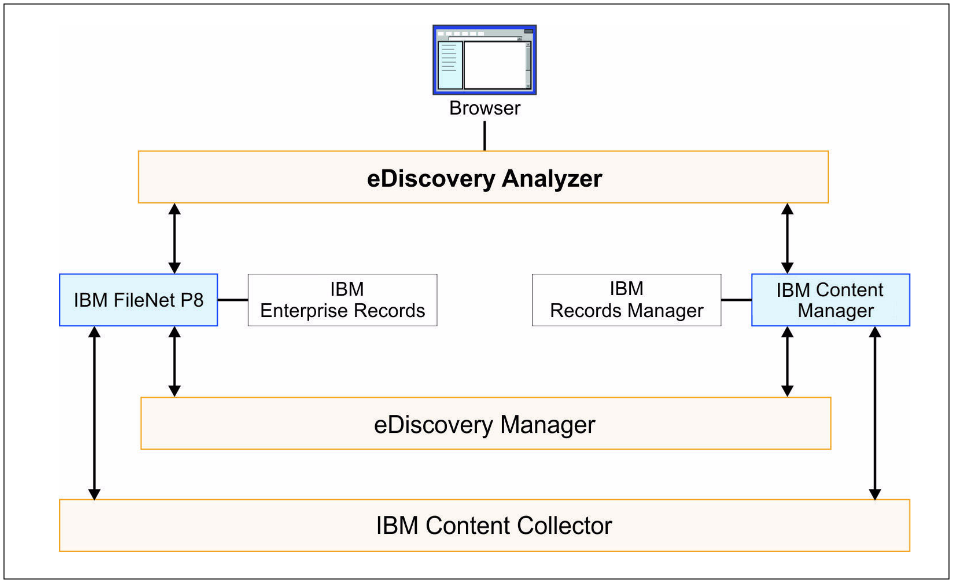


Figure 6-9 High level eDiscovery Manager and Analyzer integration architecture

6.5.3 Dependencies

eDiscovery Manager runs on AIX and Windows operating systems and requires a supported version of Websphere Application Server. eDiscovery Manager is supported in a VMware ESXi Version 3.5 virtual machine environment. For a complete list of system requirements, refer to the following web site:

http://www.ibm.com/support/docview.wss?rs=3506&context=SS8JHU&uid=swg27018713&S_CMP=rnav

6.5.4 Integration and connection points

When eDiscovery Manager is used in conjunction with IBM Enterprise Records, it is possible to declare the contents of a folder as records from the eDiscovery Manager Web client. However, with or without records management integration, eDiscovery Manager automatically places holds on those documents that are added to a case to avoid deletion in the archive server and to preserve content that might be needed.

eDiscovery Manager’s integration with Enterprise Records is optional. Documents and emails that are placed in a collection are placed on hold, and this hold supersedes any other records holds. In addition, explicit records holds can be placed directly from the eDiscovery Manager user interface.

eDiscovery Manager tightly integrates with IBM PSS Atlas eDiscovery and legal hold workflow system. IBM PSS Atlas allows attorneys to define who and what must be placed on legal hold. This information then automatically propagates these instructions to the IBM eDiscovery Manager where a case is created and information preserved.

The IBM eDiscovery Manager application programming interfaces enable other applications to integrate with eDiscovery Manager:

- REST APIs

  The eDiscovery Manager REST APIs consist of methods to manage cases and scheduled searches. The APIs enable a case to be created, eDiscovery Manager to be included in a workflow, discovery orders to be submitted, and search results saved to a case.

- Email converter APIs

  The eDiscovery Manager email converter APIs include methods to work with a variety of email formats and provide access to metadata, body text, and attachments.

- Export plug-ins APIs

  The eDiscovery Manager export plug-in APIs allow developers to create custom export plug-ins to customize each stage of the eDiscovery Manager export process.

6.5.5 eDiscovery Analyzer

IBM eDiscovery Analyzer searches email, attachments and other documents that are collected into a case by IBM eDiscovery Manager. IBM eDiscovery Analyzer is a web-based application that allows a paralegal or attorney to search, view, and analyze archived documents in the process of legal discovery.

With eDiscovery Analyzer, users can search and prepare only those cases created by eDiscovery Manager to which they are assigned. Members of the Legal team work within the broader case contents to refine searches and reduce the result set initially using literal terms against the full-text of indexed documents using keywords, date ranges, and phrases. Content can be filtered using Boolean operations and criteria against Field names and Categories. Further refinement of case matter content is performed using a variety of visual aides to graphically explore relationships between concepts, custodians, chronology, and other key elements at an aggregate level.

As this exploration and analysis takes place, users can flag documents and email messages based on whether they are responsive, confidential, privileged, require further review, and so forth. After all search and content culling activities are complete for a given case, the resulting subset of the original documents can be exported to be handed over to opposing counsel. These search methods and other operations performed by eDiscovery users are tracked in an audit log of the case, providing justification to the courts of every step of the process.

eDiscovery Analyzer interacts with eDiscovery Manager and other products, as illustrated in Figure 6-9 on page 160.

For best performance, eDiscovery Analyzer, eDiscovery Manager, and the archive server must be on separate servers.

6.5.6 Dependencies

eDiscovery Analyzer runs on AIX and Windows servers and supports a wide range of email systems and client configurations. For a complete list of system requirements, refer to the following web site:

http://www.ibm.com/support/docview.wss?rs=3533&context=SSJKLP&uid=swg27018101

6.5.7 Integration and connection points

The IBM eDiscovery Analyzer API enables other applications to integrate with eDiscovery Analyzer. As with eDiscovery Manager, eDiscovery Analyzer offers an API that conforms to REST principles. Uniform resource identifiers (URI) are used to exchange information between client applications and the eDiscovery Analyzer application server. Because REST is based on HTTP, any client or programming language that can submit an HTTP request can be used. For example, the REST APIs allow an application to index a case, create or assign flags, define ignored text, or work with search results without using the eDiscovery Analyzer client.

The eDiscovery Analyzer API consists of methods to manage cases and monitor system status, list or flag search results, and manage documents found in search results.

The eDiscovery Analyzer API is supported by a servlet on the eDiscovery Analyzer application server. This servlet processes all eDiscovery Analyzer API requests, verifies that requesters have the appropriate authority to call the APIs, and returns XML responses.

6.5.8 eDiscovery Manager and eDiscovery Analyzer summary

The IBM eDiscovery tools assist with legal discovery and internal investigations of possible violations of corporate policies by enabling the search and export of documents. eDiscovery Manager provides powerful search facilities for identifying and sequestering potentially responsive data into a case that is assigned to a set of Legal personnel for review and preparation. Using eDiscovery Analyzer, they refine the content using rich, visual tools to cluster and categorize data based on concepts, custodians, date intervals, and other logical aggregations, eliminating non-responsive, confidential, and irrelevant content from the case. These tools allow much of the eDiscovery process to be conducted using in-house personnel, providing significant savings and far better levels of preparedness for any legal matter.

6.6 Summary

IBM offers a suite of Information Lifecycle Governance solutions to help organizations succeed in meeting their legal requirements and in making better use of their information assets. These powerful products expand the IBM FileNet P8 Platform to harness the content management facilities in important areas of records management, unstructured content analysis, and legal discovery. This combination assists customers in meeting official requirements and regulations, which reduces corporate exposure and risk.

Companies collect and archive content with Content Collector, organize and classify documents with IBM Classification Module, analyze content with IBM Content Analytics, and provide records management with Enterprise Records. IBM eDiscovery Manager and Analyzer combine and expand some of this functionality to address legal concerns for search, retention, and reporting according to regulatory requirements.These products make it easier to incorporate and make use of intellectual assets. Information is better organized, more accessible, and easier to use with these solutions.

Using this tightly integrated technology stack, organizations can proactively gain control over structured and unstructured information with state-of-the-art archiving, classification, retention management, and analytics.

7

Building an ECM solution

Any solution that uses the IBM FileNet P8 Platform includes the core engines, the appropriate features of the core engines, and potentially additional add-on products. This chapter is dedicated to helping determine the appropriate software stack and how that can be used to solve a business problem. For a discussion of the infrastructure and physical implementation of the solution, see Chapter 10, “Architecting an IBM FileNet P8 solution” on page 333.

This chapter covers the following topics:

- 7.1, “Enterprise content management” on page 166
- 7.2, “Working with content” on page 166
- 7.3, “Business processes” on page 176
- 7.4, “Advanced case management” on page 182
- 7.5, “User interface” on page 183
- 7.6, “Data sources” on page 185
- 7.7, “Content processing and classification” on page 188
- 7.8, “Compliance and governance” on page 189
- 7.9, “Monitoring” on page 190

7.1 Enterprise content management

Enterprise content management within the IBM FileNet P8 Platform is gathering both content and process into a single repository. This base of content and process then becomes the foundation for other applications and uses, such as case management, forms management, or records management.

Each problem requires an appropriate combination of content, process, user interfaces, content ingestion or federation, and retrieval processes. The mix can be determined by following content or process from the start of life, through classification, retrieval, retention, and destruction. An ECM solution, shown in Figure 7-1, must define how users will interact with content from how it is found, retrieved, used, tied to business processes, retained, and disposed.

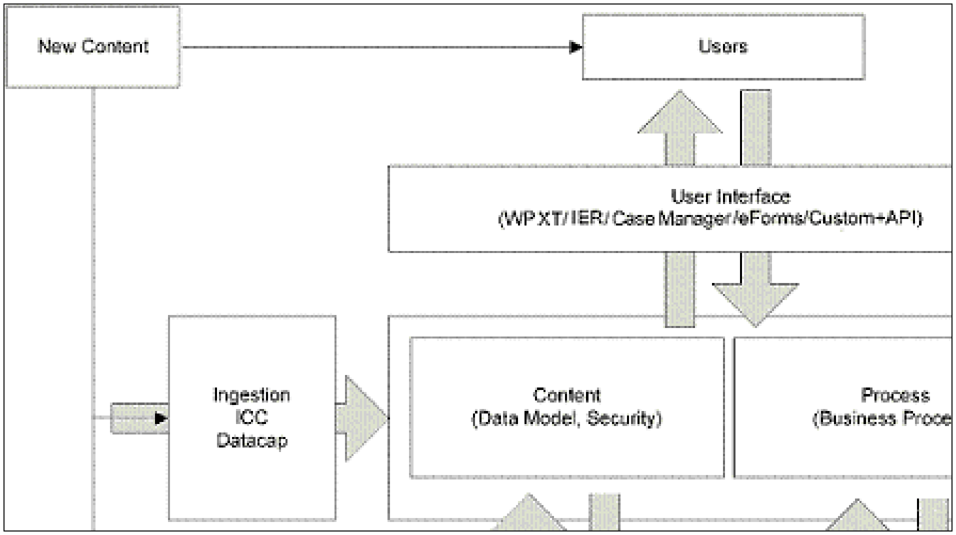


Figure 7-1 Following content through and ECM solution

7.2 Working with content

Many problems start with a document or relate to objects and content. Content is stored in the Content Engine and must be made available to users or processes to further the business goals. The purpose of this section is to show how the basic elements of content management, described in 2.2.1, "Data model" on page 23, are used to build an ECM solution. For best practices and recommendations, see *IBM FileNet Content Manager Implementation Best Practices and Recommendations,* SG24-7547.

7.2.1 Data model

The data model exists to help users and applications find content applicable to a business problem. While some applications and users can access content directly by the unique identifier, most of the time searching for and finding content relies on metadata.

Metadata

In any content-centric or content-based application, there must be a way to find and access content. In a file system-based application, documents are typically located by some absolute reference, such as file path. If a user is to retrieve that document, the user must know where the file exists. Similarly, in an ECM application, each object has a unique identifier or a way to directly access the given object. Within the Content Engine, each object has a unique identifier, stored as a 128-bit GUID.

If the user does not know the ID or if it is not stored in a readily accessible place, there must be a way to search for the object in question. To allow for efficient searches, content that is stored in a repository must be accompanied with additional information that allows finding particular objects or distinguishing different pieces of content that were stored. For this purpose, *metadata* must cover common attributes that are maintained at system level (for example, who added a certain document to the system and when it occurred) for all objects and individual information, which is only used for a limited group of objects that belong to the same type (a vendor number for invoices).

In addition to system-level metadata, the objects stored in an ECM system must also be assigned metadata that is appropriate to the given application. This metadata can include a customer or account number, case ID, employee number, or project name. After the document is stored in the system, users and applications can utilize this metadata to help present the user with the correct objects and information to assist in completing their business tasks.

All content within the Content Engine exists as objects. Every object in the system has similar characteristics and metadata that can include:

- Unique ID (a 128-bit GUID)
- Metadata describing the object (including date created, modified)
- Security (for securable objects)
- Storage policy and storage area (for objects with content)
- Lifecycle state and version (for versionable objects)

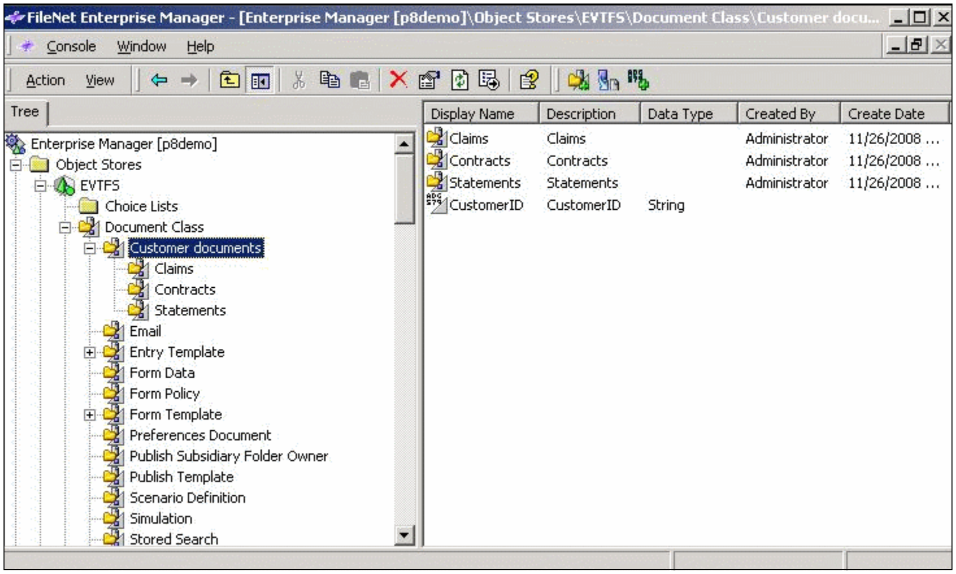
Classes

Within an ECM context, it is the class that helps define what type of content is being stored and is the base level of differentiation that can be applied to a given object. The class structure also defines the taxonomy or metadata structure within the repository. Classes define:

- Metadata collected on an object
- Storage area or policy
- Security
- Life cycle

While the class might define the defaults, any application can override the defaults, such as security, at creation time. Some of the items can be modified after an object is created, assuming the user has the appropriate security rights to do so.

Because IBM FileNet P8 is an object-oriented content management system, classes have a hierarchy. Just like in Java or other object-oriented languages, this hierarchy allows for inheritance of properties and actions. Figure 7-2 shows a sample document class hierarchy.


FileNet Enterprise Manager - [Enterprise Manager [p8demo]\Object Stores\EVTFS\Document Class\Customer docu...
Console
Window
Help
Action
View
Tree

- Enterprise Manager [p8demo]
  - Object Stores
    - EVTFS
      - Choice Lists
      - Document Class
        - Customer documents
          - Claims
          - Contracts
          - Statements
        - Email
        - Entry Template
        - Form Data
        - Form Policy
        - Form Template
        - Preferences Document
        - Publish Subsidiary Folder Owner
        - Publish Template
        - Scenario Definition
        - Simulation
        - Stored Search

Display Name	Description	Data Type	Created By	Create Date
Claims	Claims		Administrator	11/26/2008 ...
Contracts	Contracts		Administrator	11/26/2008 ...
Statements	Statements		Administrator	11/26/2008 ...
CustomerID	CustomerID	String		

Figure 7-2 Document class hierarchy sample

Properties

As metadata is important to an ECM system, the Content Engine provides for a way to define and store metadata through properties. Each class has a number of properties assigned to it. These properties define what metadata to collect and of what type, for example, an object that refers back to an account can use an alpha-numeric account number, which is then stored in a single-valued string property. In the case of an email message, there might be one or more recipients, so these can be stored in a multi-valued string property.

When defining a metadata taxonomy within IBM FileNet P8, the administrators, developers, or business analysts determine what metadata must be collected to assist in finding the objects required for a business process and then ensure that classes are created to collect the appropriate information.

As part of defining the properties to be added to the class, it is necessary to determine not only the type of data, determining the data type, but also things, such as:

- Is a value required for the property
- Is there a definitive list of possible properties, see “Choice lists”
- Are there properties that should be hidden from users
- Should the value be changeable after the object has been created

The global property template defines the default values for each of those settings. If no changes need to be made at the class level, the properties can be assigned directly to the class. Some of the configuration settings can be overridden at the class level, allowing for additional refining on a class-by-class basis.

Choice lists

When collecting metadata, there might be a property that has a strict set of values, such as state or province. While this value can be constrained within an application, the Content Engine can also provide a validation mechanism through a *choice list*, which is a set of possible string or integer values for a given property. When stored in the Content Engine, the value of a property that is constrained with a choice list must be in that list, or the Content Engine will not allow that object to be stored.

A choice list can also be a nested list of values. This nesting allows some grouping of values to help simplify finding the appropriate value in a larger list. Workplace XT and other provided applications can present the defined list of items to the user. The API also allows for custom applications to retrieve the choices for display in the application.

Documents

Objects with content, versions, or life cycle are stored in the class *Document* or one of its subclasses. Objects of this type store the metadata and the information about the location of the associated content in the repository. Instances of document class are *versionable*, which means that when a new major or minor version is created, the information about the content of the existing version is kept and can be reused for later purposes. It is not required that the content element is stored in the repository because the Content Engine supports the access to the content through federation, UNC, and URL. In fact, the existence of content is not enforced. Objects can be created without having content associated.

When a new object store is initialized, several subclasses can be automatically created using add-ons, for example, IBM FileNet P8 applications, such as Workplace XT, use a subclass called *Preference Document* to store user and site-related information as XML files in the Content Engine. Another example is the data documents created by eForms, which are also stored underneath a subclass of the document class. With the enterprise perspective in mind, organizations will define all of the properties that documents need to share across the enterprise at the level of the foundation class and certain security information, such as read access to auditors.

Objects in the document class hierarchy can have zero or more content elements. A document with a single content element can be a single office document or a single image. Multiple content elements can be useful if a document contains multiple pages in multiple files, such as a single document composed of multiple scanned pages. This type of element is helpful especially when pages might need to be rearranged at a later time or when single pages must be displayed to the user at viewing time, so page caching is faster, which facilitates faster document viewing. To add further content elements, the document must be checked out, and after all content elements are added, it can be checked in again. Each version of the document can have a different number of content elements assigned.

Content Engine also offers a way to link multiple objects together to handle compound documents or documents that are actually built using a number of other documents. See “Link objects and compound documents” on page 172.

As content sometimes needs to be commented on without making changes to the underlying content, the Content Engine offers an additional capability to annotate content. Annotations are notes or comments about a given piece of content that can be used by applications and displayed to users. One common use of annotations is to store comments or notes on an image that is displayed to users by the viewer in Workplace XT.

Custom objects

In addition to objects with content, there are use cases where it is helpful if the content management system can handle objects that only have metadata. The Content Engine uses the class *Custom Object* for this purpose. Custom object instances differ from document instances in three ways: custom objects do not have content, do not have a life cycle, and are not versionable. If versioning for objects without content is needed, the appropriate object class can be defined in the Document class hierarchy.

Custom objects are often used to model the *business objects* that are required for an application based on the ECM platform. There are several benefits to using custom objects instead of storing this information outside of the repository in a separate database:

- The same access control principles are imposed on custom objects that are imposed on content objects.
- Custom objects support events and auditing for more details.
- Custom objects can participate in business processes or records management just as normal documents can do.
- Custom objects can be related easily to other objects that are maintained in the Content Engine utilizing Link Objects.

Foldering

The *Folder* class hierarchy allows modelling folder structures that users are often familiar with when working with documents. This process can be useful because users are accustomed to folders from their work with any file system.

A folder in the Content Engine shares many properties of a folder in a file system in that it is a container into which other objects (including other folders) can be filed and that it is access control enforced. It is possible to determine which security entities (users or groups) are allowed to file objects into a folder or are allowed to create subfolders. An object can be filed into any number of folders in the Content Engine and it is easily possible to obtain a list of all of the folders that an object is filed into.

Most importantly, events also apply to any folder instance in the Content Engine, which means that events for the folder can be generated by the Content Engine, for example if a new document is filed into it. Folders also have properties and the folder class hierarchy can be used to inherit the properties from parents to children. Folder properties allow efficient searches for folders compared to traditional browsing access only.

Folders are commonly used to structure content elements in some context, for example as a *case folder* that contains documents that belong to a certain case or a customer file folder structure that imitates the file plan that might have existed for paper-based documents.

Additionally, documents can inherit their security from folders through the SecurityFolder property. See Chapter 8, “Security” on page 191 for more information about security inheritance.

Link objects and compound documents

Objects are often related to each other and these relationships help to structure and efficiently manage objects in daily life. The same applies to objects that are stored in a content management repository. Folders are a way to express a relationship and hierarchy between different objects that somehow belong to each other, but the Content Engine offers another tool for creating and maintaining relationships, called a *link* object.

A link object has a *head* and a *tail* property of type *object* that determines the two objects that are related to each other. There is a one-to-one relationship, which means that if one object relates to *N* other objects, *N* link objects must be created to express this condition. Link objects have properties and security, which means that different users might or might not see specific links. The link class can be subclassed to restrict the object classes, which can be linked together by the new subclass.

In addition to generic links, the Content Engine provides a compound document framework. *Compound documents* can represent a complex relationship between objects for content that is by definition a single unit. In contrast to link objects, using the compound document framework you can also express complete hierarchies of relations, which is done based on *Component Relationship Objects*, which can be subclassed if required. Component Relationship Objects have a parent and a child, and any child can again be a parent for one or many other relations. The Content Engine provides efficient methods to explore the hierarchy of a compound document that is built on Component Relationship Objects. The compound document framework can also be used to maintain a document and its renditions into several formats or translations into various languages, for example.

There are important differences between link objects and Component Relationship Objects to express relationships:

- Link Objects have their own security, whereas Component Relationship Objects inherit security from the head object.
- Link Objects always point to a fixed version. The child of a Component Relationship Object can be configured to point to either a fixed version, the latest version or latest major version of the corresponding version series, or even to the latest version or latest major version that has a certain label value.

These flexible options allow the compound document framework to express complex inter-object relationships that are found in areas, such as technical documentation.

7.2.2 Document life cycle

Content-based objects, such as documents that are frequently subject to a life cycle that spans from the initial version of this document up to a final release, might need to be kept for a defined retention period. Even for the simpler case of storing images of scanned documents that are processed in a workflow, there might be a change in the documents' life cycle after the processing completes, and the document must be stored until its final disposition date. The Content Engine supports the definition of the document life cycle and life cycle events that drive the document life cycle.

In the Content Engine a document life cycle consists of these objects:

- A *lifecycle policy*, which defines the stages of the life cycle for this document
- The *lifecycle event*, which fires when the document changes from one stage to another
- The *lifecycle action*, which is linked to the event and can be used to perform custom operations

Lifecycle policy

A lifecycle policy can be inherited by a document through a definition at the level of the corresponding document class, or it can be assigned to the document at creation time. As described earlier, using the Content Engine you can override the inherited lifecycle policy to a different one, if required.

A lifecycle policy can contain permission information, which is automatically applied to the document when the life cycle state changes. The *changeState* method for the document must be called to change the state for the document, which can promote the document to the next state, demote it to the previous state, reset to the initial state, or set the document into an exception state that temporarily blocks the document for further transitions until the exception state is cleared.

Refer to 8.4, "Setting security across the enterprise" on page 229 for more details about how lifecycle policies and security policies can be used to manipulate the permission control for objects in the Content Engine.

Lifecycle action

Lifecycle actions can be compared to custom event actions. They allow you to extend the capabilities of the Content Engine by assigning a Java class that is called to handle the lifecycle event.

7.2.3 Security

Controlling the access to content is crucial for an ECM system. It is mandatory to impose access control to the objects maintained in an ECM repository to ensure that the appropriate information can only be accessed by the correct audience.

Security within the Content Engine is applied on the individual objects as an Access Control List (ACL). Each ACL has a number of Access Control Entries (ACE) that define the security principal, the specific rights, and whether those rights are allowed or denied. A principal is a user or group that is contained in the configured LDAP security provider and is linked to in the given ACE.

In addition to ACL-based security, the Content Engine also offers other ways of defining and delegating security including: marking sets, security inheritance, and security templates.

For a complete look at security within the Content Engine and how it applies to content, see section 8.2.1, “Content Engine” on page 208.

7.2.4 Storage

When architecting an IBM FileNet P8 solution, it is necessary to determine the type of storage and how the storage can change through the life cycle of the content. Some factors that help identify storage requirements are cost, speed, availability, and retention.

Simple storage

As described in section 2.2.5, “Content storage” on page 31, the Content Engine has built-in support for both local and network attached storage (SAN or NAS) and support for storing content in the same database as the metadata. For instances where there is no need for hardware retention or content addressable storage, these storage methods can be used to efficiently store content.

Hardware retention and WORM

There are some situations where either legal requirements or internal standards dictate that content be stored in a medium that facilitates hardware-based retention. In these situations, a device, such as the IBM Information Archive, might be appropriate because they provide hardware-based content retention that the Content Engine will honor. For more information, see “Fixed Content Device (FCD) storage” on page 33.

Hierarchical storage

In some cases, it is desirable that the life cycle of a document has consequences on the location where the document is stored, for example, a contract document that is created by iterating through several negotiation cycles, each represented by a new version of the document. After the content is finalized and approved from both sides, the latest version of the document, or a scanned image with the signatures on it added as the latest version, it might be desired to store this version on a fixed content device that implements WORM storage.

If required, this concept can be extended to implement content aware storage, where the storage location for content is shifted based on the current state of the content within a workflow or its lifecycle, for example from expensive to cheap storage or to a fixed storage area. This is another example that demonstrates the flexibility of the IBM FileNet P8 architecture because it allows an implementation that is superior over the capabilities of other ECM systems where the storage subsystem itself decides to move content to different storage locations because this is done only based on fixed retention times that were defined.

Federation

Of course, since the Content Engine supports content federation out-of-the-box, it is possible to have content stored in a mature or third-party system, such as IBM FileNet Image Services. For more information about federation, see Chapter 5, “Expansion products for connection and federation” on page 117.

7.2.5 Classification

After the data model is defined, it is necessary to determine the appropriate method for classifying documents and applying metadata. There are three basic ways to complete content classification: manual or user-based classification, semi-automated classification using Capture or Datacap, and automated classification with the Content Engine classification framework or the IBM Classification Manager.

Manual classification

The first way content can be classified is by requiring the users who create the content to apply the appropriate metadata. In the case of Workplace XT or FileNet Integration for Microsoft Office, this occurs either through using wizards or entry templates. These wizards and entry templates prompt the user for the target folder (if applicable), class selection, property entry, and content selection. Depending on how the properties are defined for the class, Workplace XT will allow for single or multiple values, selection of values from a choice list, or leaving the property with no value.

For instances where business analysts or administrators want to remove options from the user, Workplace XT offers entry templates as a way to predefine folders, security, and even property values.

For more details about content creation and manual classification with Workplace XT, see 3.1.2, “Creating content” on page 66.

Semi-automated using Capture or Datacap

A second way to classify and apply metadata is using Capture or Datacap. Each of these tools offer abilities to derive metadata values from the actual content of the paper or documents being scanned and ingested. These capabilities include deciphering bar codes and scan code, optical character recognition, and zone-based data extraction. These capabilities can be leveraged in addition to prompting the user to enter the appropriate metadata as part of the capture process.

For more information about Capture and Datacap, see section 4.4, “IBM Datacap” on page 101 and section 4.5, “IBM FileNet Capture” on page 109.

Automated classification

The third way to apply metadata and classification is to use automated classification tools. The Content Engine provides an extensible classification framework, including an XML classification tool. See section 2.2.8, “Classification” on page 38 for more details.

In addition to the Content Engine classification framework, the IBM FileNet P8 Platform also integrates with the IBM Content Analytics and IBM Classification Module tools. These tools can work with the content of the documents themselves to help discover metadata and classification information and then apply that to the documents and objects within the Content Engine.

For more information about Content Analytics and Classification Module, see section 6.3, “IBM Classification Module” on page 144 and section 6.4, “IBM Content Analytics” on page 151.

7.3 Business processes

A business problem often has a process component to it. Whether it is a credit card application that must be processed, a case that must be worked, or a user request that must be handled, there is typically a defined process to complete the task. Business process management using IBM FileNet P8 BPM and the Process Engine provides the framework for building a process management solution through workflow design, routing, application of business rules, and integration with the content stored in the Content Engine.

7.3.1 Workflow definitions

For IBM FileNet P8 purposes, workflow or business processes typically mirror the steps to solve a business problem. The workflow defines who needs to do what with the information, when it must happen, and how. This definition is defined in a workflow map that is launched by some initiating event, either by content creation or by a process-centric application.

In migrating an existing business process to the IBM FileNet P8 Platform, first, identify if the process can be implemented using Case Manager and its task-based routing. See 7.4, “Advanced case management” on page 182 for more information.

If Case Manager is not appropriate, it is necessary to determine what the steps of the process are and then for each step, identify:

- What needs to happen for that step to complete
- Who or what needs to work each step
- When it needs to happen

After determining this information, the workflow definitions can be created in Process Engine.

What

The element of a workflow is what happens during that step of the workflow. For a Process Engine workflow, there are four basic methods of processing work: human-centric step processors and UIs, the Component Integrator, process orchestration, and custom work performers.

Human interaction

Content-centric processes must provide access to content objects that are stored in the repository, and they must support the creation of new content in the repository as part of the business process. The options that are available to accomplish this are:

- Default HTML step processors

  The IBM FileNet P8 Web client Workplace XT, allows the user to access his or her personal inbox and any other public work object queue, based on the permissions granted. When a user opens a process instance (or work object), it is represented in a step processor, which is an HTML page. This step processor allows the user to view the attached documents (assuming that the user has the appropriate permissions) and add new content to the process instance, if the attachment field is configured for writing access.

- FileNet eForms and Lotus Forms

  In many use cases, business processes are driven by forms or a form-like representation of data. To address this need, the IBM FileNet P8 Platform offers eForms, an HTML representation of a form that was designed using the forms designer tool. eForms can be displayed in any Web browser that IBM FileNet P8 supports without needing to install additional software on the client PC.

  Using the Forms Integration Framework you can use Lotus forms in the same way that you use FileNet eForms.

  For more information about electronic forms, see section 3.3, “Electronic forms” on page 68.

- Business Process Framework

  The Business Process Framework (BPF) focuses on case management use cases and provides a highly-configurable user interface for this purpose. Like eForms, BPF tightly integrates with the Content and Process Engine and allows integrated access to content objects that are part of the case. Based on the permission of the user, new content elements can be added to the case or existing documents can be replaced by a new version. BPF also enables users to add files from the local file system to a case that automatically adds the content to the repository as well. For more information about BPF, see section 3.4, “Business Process Framework” on page 74.

- API-based step processors

  If the out-of-the-box capabilities are not sufficient, custom step processors can be implemented to act as the user interface to the process, for example, external applications can be integrated seamlessly into the user interface, or a thick client can be implemented. A custom step processor will use the Content and Process Engine API to access process data and content information and information from external data sources or applications and represent them the way that all requirements are met. As the logic of business process is defined on the level of the Process Engine workflow definition, it can be altered to meet changing requirements without changing the user interface.

  For additional information about developing with the APIs, refer to *Developing Applications with the IBM FileNet P8 APIs*, SG24-7743.

- Widgets

  IBM FileNet P8 features widgets that allow changing the user interfaces rapidly. Consider a widget as an area in the browser window. A designer can define the user interface by placing the widgets in a design application at the desired position inside of the browser window.

The widgets follow a common specification, which allows them to communicate to each other and to exchange data between each other, for example:

- A process data field widget that has the zip code as one of its data fields to pass this information to a map widget, which then automatically displays an area map for this zip code.
- A customer number from the work object data widget that is passed to a master data widget that collects and displays additional information for this customer, such as name or address.

This approach makes user interfaces to processes even more flexible and allows the reuse of widgets across different applications. For more information about Widgets, see section 3.5, “ECM Widgets” on page 82.

Component Manager

The Component Manager provides a way to call custom Java classes from a step on a workflow map. This allows data to be passed from the process instance to the Java component and retrieving the result back into the process instance. The IBM FileNet P8 Platform ships with one component (Content Engine Operations), which allows interaction with the Content Engine to modify content objects.

Custom components can perform virtually any operation and facilitate the interaction with external systems. For instance, if an architecture heavily relies on Java Messaging Queues (JMS), such a component can be used to read and write messages from JMS queues. Because the components are written in Java, existing business logic can be reused.

The Component Manager uses the flexibility and the open architecture of the IBM FileNet P8 Platform to allow a direct integration with external systems at the level of a workflow step.

Process orchestration

Complex business processes involve operations on many different systems throughout their execution. In many cases, parts of the process might already be implemented in a given workflow environment. Instead of re-implementing the complete process on a new BPM system, only the basic process flow is modeled on this BPM system. The process is then *orchestrated* by calling functionality from an existing system or even starting a process on another workflow system and waiting until the result is returned so that the main process flow continues.

This process allows leveraging the existing assets and capabilities of the systems that participate in the business process. For example, the

Content-centric part of a process might only be an exception route of the main process, which itself is running on IBM WebSphere Process Server because it heavily involves the transformation of business objects that are defined for different systems into each other. Staying in WebSphere Process Server requires implementing a human task with integration into the IBM FileNet P8 Content Engine repository to display a document that the user needs to make a decision, how to resolve the process exception. Instead, the exception process in FileNet BPM can be launched from the WebSphere Process Server engine, passing the reference to the document. The Content-centric exception process can take advantage of the tight integration into the Content Engine to display the document (and possibly other related content objects) and the FileNet BPM process sends the result back to the WebSphere Process Server process, which continues its execution.

For more information about process orchestration, see section 2.3.4, "Process orchestration" on page 46.

Custom work performers

Custom work performers are the most flexible option to enhance the capabilities of the Process Engine or to integrate processes with external systems. A custom work performer can be considered as a background process that regularly polls a queue on the Process Engine for new work. Custom work performers use the Process Engine API to query the queue, lock a work object, read data, perform the configured operation, write data back to the work object (optionally), and dispatch it to the next step.

Compared to the solution of implementing a Java class and using the Component Manager, a custom work performer can be implemented as an executable, a service, or a demon process that allows, for example, easier access to system resources, compared to the Java class, which is executed within the same Java Runtime Environment (JRE) as the Component Manager.

Who

Any business problem typically has steps that must happen to complete the process. Those steps must be assigned to the appropriate person, work performer, or external system to process that step. Within the Process Engine, there are several mechanisms for directing work:

- User queues direct the specific work step at a user or a group.
- Public queues allow work items to fall into a publicly available queue so users and groups are able to pull from a central queue of work to process.

- In-baskets allow users to view work though in-baskets as a way to further restrict or reduce the number of items for a user to process.
- Component queues direct work to the component manager for external processing. See “Component Manager” on page 179.

When

After each step of the workflow, it is necessary to determine the next appropriate location or user to further the business process along. The Process Engine has complex routing and conditional branching capabilities to help facilitate these types of interactions.

Map-based routing

Workflow maps can have a number of routes and at the end of each step, there are places to do either direct or conditional routing. Conditional routing allows the assignment of the work based on the properties or attributes of a given workflow.

Business rules engine-based routing

While conditional routing can be accomplished through a workflow map, there is a drawback. After a workflow map is built, it is transferred and compiled in the Process Engine. Any routing logic changes require the workflow to be transferred again and will not affect any existing workflows.

A Business Rules Engine (BRE) is one flexible way to handle this problem. The business rules are stored centrally in the BRE, which provides interfaces so that arbitrary applications can then use the rules. In addition, many BREs support the definition of the rules in a user friendly business vocabulary, which makes it much easier for the business units to define and maintain the rules in the BRE. Another important benefit of using a BRE is that it allows enforcement of consistent rules across applications because the rules are defined only in one place and re-used from different applications. This process matches the idea of re-usability.

The Process Engine supports using external business rules in a process by calling BREs from a process at runtime. The rule name and the parameters to pass are configured at the process definition level. At execution time, the Process Engine calls the BRE using a Rules Engine Framework. The BRE evaluates the rule and sends the result back to the Process Engine. Based on the result provided, the Process Engine dispatches the process instances into the appropriate route. The Rules Engine Framework plugs in BREs such as ILOG®.

By using a BRE, the processes that are executed on the Process Engine gain the flexibility that conditional routing can be performed based on business rules that are centrally stored and maintained by business units. BREs are helpful in Content-centric processes because they can be used to define the routing of documents to in-baskets on group or individual levels, externally, to the process definition. This ability improves the agility of business processes because they can immediately be adjusted to changing conditions in the way automated decisions are made in the process.

7.3.2 Task-based processes

In addition to explicit, workflow map-based business processes, the addition of Advanced Case Management enables an additional approach to solving a business process. This approach is task-based instead of strict-assignment based. Task-based workflows allow more flexibility in completing work assignments by breaking them down into tasks that can be completed as appropriate, involving the appropriate resources. For more information, see section 7.4, “Advanced case management” on page 182.

7.3.3 Analysis and optimization

While running process-intensive business solutions, it might be necessary to perform analysis on currently running workflows and historical performance to detect trends and suggest optimizations that can be made to reduce processing time. IBM FileNet P8 BPM offers two tools to help with these tasks.

Case Analyzer

The Case Analyzer will work with the Process Engine event logs to generate data that can be used to help analyze past performance. This performance data can help identify bottlenecks and determine how recent and past changes affected performance times.

Process Simulator

The Process Simulator can work with the past data and help determine which changes to the process definition must be applied to eliminate given inefficiencies or bottlenecks.

For more information about the Case Analyzer and Process Simulator, see section 2.3.6, “Analysis and optimization” on page 48.

7.4 Advanced case management

For many, content and process management, the item being stored and modeled, is not just a document or a simple process, but instead can be thought of as a case. Examples of a case are a collection of documents that are patient records, the materials involved in a lawsuit, and documents relevant to an insurance claim. No one document stands alone, and to complete the business problem or task requires having all of the related documents grouped together and available to the user.

The second aspect of this type of business process is that it does not directly lend itself to rigid process structure. Instead, the process itself is only semi-structured and the user working the process needs some flexibility from the rigid system process to complete the work item.

To facilitate this type of work scenario, IBM FileNet P8 5.0 adds Case Manager strives to bring together the content, process, business rules, people, and history necessary to handle managing that work item.

Case Manager is implemented on top of the existing platform and offers:

- A platform for designing and building case solutions
- A runtime environment for launching, processing, and interacting with cases
- A set of tools for configuring and moving solutions into production environments
- A set of APIs and templates for customizing case solutions

A case management solution is built using the following objects:

- Case: a set of related activities, content, processes, and collaboration artifacts used together to manage a specific business activities.
- Case Type: a definition of a case
- Task: A list of items that need to be completed in the context of a case (i.e., a to do list).
- Role: a collection of user accounts associated with a specific business function. These are used to access a particular task or work queue step.
- Document Type: a document class in the P8 Content Engine.

In addition to the data types, Case Manager heavily utilizes the roles, in-baskets, and queues within the Process Engine. These allow Case Manager to deliver the flexibility necessary to enable users the flexibility to accomplish their responsibilities in the way that makes the most sense for the task or tasks they have been assigned.

7.5 User interface

Enterprise Content Management is often user focused and user driven. This requires users to interact with the system to solve business objectives. Selecting the appropriate user interface can be important to properly presenting the appropriate information to the appropriate user so they can complete their tasks. An ECM solution can rely on one or more UIs, including both those available as out of the box applications as well as custom applications built against the IBM FileNet P8 APIs.

7.5.1 Out-of-the-box

The IBM FileNet P8 Platform offers several out-of-the-box and optional user interfaces to handle common business problems. These user interfaces and applications can often be sufficient to handle the business tasks and requirements of a given solution.

Workplace and Workplace XT

Workplace and Workplace XT are web applications that you can use to access content and process engines. They offer similar functionalities but have some differences. Workplace XT is the new and preferred interface. It works with IBM Case Manager, eForm, and ECM Widgets. Workplace works with eForm, Business Process Framework, and is customizable. There are no new features added to the existing Workplace. Use Workplace only if you already have it or you have business needs for features that are not available in Workplace XT.

Workplace XT handles many business needs including the ability to create, retrieve, update, and delete content. Workplace XT also provides a standard set of step processors for working business process and direct access to a users inboxes and the public queues.

Because Workplace XT can also be where users add content to the system, it offers a number of ways to do so. These are described in 3.1.2, “Creating content” on page 66. One key feature that Workplace XT has for helping users create content is the entry template. When configured, an entry template can simplify the creation process by both hiding steps and predefining values such as target folder or security. This means a user only needs to select or enter the values that directly apply or that only they can supply.

Electronic form

Some business problems involve gathering and processing information that is forms based. eForms and Lotus Forms can be used to display forms to the user and lock them to business processes. See 3.3, “Electronic forms” on page 68.

Records management

For solutions that require compliance, the IBM FileNet P8 Platform supports IBM Enterprise Records for records management. In addition to the underlying compliance layer, Enterprise Records helps manage compliance and records through a Web-based interface. For more information, see section 6.2, “IBM Enterprise Records” on page 137.

Case management

For case-based lines of business or solutions, IBM offers functionality in Case Manager that will help with gathering together the elements and supporting materials for a case. It also offers a task-based system for assigning tasks to appropriate users or groups within an organization. 7.4, “Advanced case management” on page 182.

FileNet integration for Microsoft Office

If the solution requires content creation through the Microsoft Office suite, the IBM FileNet P8 Platform has integration with the Office suite to allow users to create content directly from Office applications.

Custom user interfaces

There might be instances where the stock applications are insufficient for a given task. The APIs offered within the IBM FileNet P8 Platform offer a rich set of tools for accessing both content and process. When coupled with modern UI toolkits, they can be used to build user interfaces customized to the specific needs of a line of business.

For additional information about developing with the IBM FileNet P8 APIs, refer to *Developing Applications with the IBM FileNet P8 APIs*, SG24-7743.

7.6 Data sources

In an ECM solution, content can come from a number of sources, including users through applications, such as Workplace XT, as described in “Workplace and Workplace XT” on page 184. However, there might be a need for large volumes of content to be loaded into the system on a regular basis. This content can come from external systems, be paper based, or be electronic content. For these needs, the IBM FileNet P8 Platform and add-on tools offer a number of solutions.

7.6.1 Ingestion

If content is not directly generated by users, such as through a word processor or some other content generation tool, it must be brought into the system, which is often called content ingestion. Ingestion brings the content into the system by creating new objects, attaching the content, and applying metadata. The IBM

FileNet P8 Platform has two basic tool types for the task: ingestion tools or scan and capture tools.

IBM Content Collector

For content that starts as either files on a file system, content in an email repository, or in Microsoft SharePoint, the Content Collector can ingest the content into IBM FileNet P8 for using in business processes or for compliance. Content Collector provides a rules-based content ingestion framework that is directly integrated into the IBM FileNet P8 Platform.

For more information about the specific capabilities and versions of Content Collector, see section 4.3, "IBM Content Collector" on page 94.

Scanning

If content to be ingested is not electronic, it must be digitized by scanning for storage and retrieval. One common usage for scanning is to keep electronic copies of cancelled checks for future reference. There are two primary ways to import scanned images into the Content Engine: direct scanning using Capture or Datacap and having a third party scan the images for import using Content Collector for Files. For scanning, review section 4.4, "IBM Datacap" on page 101 and section 4.5.3, "IBM FileNet Capture products overview" on page 112 to determine the appropriate scanning solution.

7.6.2 Federation

If content already exists in an established or third-party content management system, such as IBM FileNet Image Services, it might be desirable to standardize on a single enterprise system or catalog that acts as the final repository of information. This repository is then responsible for facilitating user access, security, retention, and disposition. To facilitate this, IBM FileNet P8 offers *content federation services* (CFS) as an integral part of its architecture. The content federation layer in the Content Engine provides three important functions:

- It queries the third-party repository for new content, retrieves the metadata for that content, maps it according to the configuration, and creates a new object in the Content Engine with that metadata and the information of where the content itself is stored, which is a unique identifier for that content in the third-party repository.
- It queries the third-party repository for any updated content or metadata for objects that are already federated and updates the Content Engine information accordingly.

- When a client sends a request to retrieve the content to the Content Engine, it uses the appropriate CFS content connector to retrieve the content element from the third-party repository and delivers it to the client.

Content federation offers important benefits:

- Events are seamlessly applied to the federated content.
- For any client, federated content is fully transparent, which means that there is no difference between content that is stored in the Content Engine or content that is federated into it.
- Federated content can therefore participate in any business process that is executed on the Process Engine and is automatically enabled for records management.
- Content Engine security can be applied to get a unified security model across all content that is under management.
- A unified taxonomy for content can be implemented on the level of the Content Engine, which makes it easier to find content throughout the enterprise.
- Content Engine and its sophisticated APIs can be used to implement new solutions that need to access content, which allows the implementation of a single source of truth of content within the enterprise. Together with the SOA capabilities of the IBM FileNet P8 Platform, this enables the customers to implement new agile applications that can easily adjust to changing conditions.
- Content federation allows a smooth transition if desired. Existing functionality can be implemented step-by-step on top of the IBM FileNet P8 APIs and can reuse the existing content in the third-party repository. No big bang migration is required because some applications can still work and change (or even add) content to the third-party repository, whereas other applications are already using the federated access through IBM FileNet P8. New content can then be created in the Content Engine and existing content can be moved behind the scene from the third-party repository to the Content Engine.

The federation system within Content Engine has two primary methods of accessing data: direct from Image Services and from other repositories using IBM Content Integrator. For more details, see Chapter 5, “Expansion products for connection and federation” on page 117.

7.7 Content processing and classification

After content is ingested or federated into the IBM FileNet P8 content repository, the business problem might require additional content processing. This content processing can include content classification, event processing, and renditioning.

7.7.1 Classification

The IBM Classification Module is a product that focuses on the categorization of content. Based on a given taxonomy, it performs an analysis of the content and proposes the taxonomy entry with the best fit, for instance, a document class and a folder. Classification Module can be used to automatically assign metadata, such as the document class or a folder, in the Content Engine when the document is added.

Classification Module can be integrated into the automated archival process that the Content Collector manages, which enforces the usage of a common taxonomy and therefore a common classification for important metadata, such as the document class. Classification by Classification Module can also be launched as a post processing operation after the content is added to the Content Engine by employing active content.

For more information about classification tools, see section 6.3, “IBM Classification Module” on page 144 and section 6.4, “IBM Content Analytics” on page 151.

7.7.2 Events

Another way to interact with content in the system is to use the Content Engine’s ability to execute code when certain events are executed. These events can include the creation of a new object or the change of metadata for an existing object. For each event, multiple actions can be defined and for each of them it is possible to define filter conditions that further narrow down whether the action is taken. Filter conditions can be based on all kinds of metadata that is related to the object that fired the event. An event can trigger an action, Java code that is executed in the context of the Content Engine, or launch a workflow in the Process Engine.

For more information about Events, see section 2.2.3, “Event framework” on page 27, and refer to *Developing Applications with the IBM FileNet P8 APIs*, SG24-7743 for additional information about developing event handlers.

7.7.3 Renditioning

On additional way to process or change content that has been created in the Content Engine is to rendition that content. Renditioning works by converting the original content into a different format, such as PDF or HTML. This renditioning allows the original document to remain in the system for later reference or refining, but creates a non-editable PDF or cross platform HTML document. These new versions can be deployed with different security or to a different section in the repository. For more information, see section 2.2.7, “Publishing” on page 37.

7.8 Compliance and governance

With the explosion of electronic content and the need to both handle the content from a records and legal discovery process, it might be necessary to implement both an electronic records system and a form of electronic discovery.

7.8.1 Records management

A record is any type of content stating results achieved, pertaining to, and providing evidence of activities performed. A record has the following characteristics:

- Fixed content in either physical paper format or in electronic format
- Evidence of a transaction, activity, or fact that has legal or business value
- Specific retention period based on company policy and regulatory rules
- Owned by the company, enterprise, or government

Records management typically involves support for the following operations:

- Defining a file plan to store records
- Identifying the information that needs to be declared as record
- Categorizing the records
- Retaining records for a specific period of time
- Destroying records when an organization is no longer obliged to retain them
- Preserving an audit trail of all activities related to the records

Two key factors in records management are:

- Records preservation: Ensure that records are maintained and accessible until the appropriate retention period elapses

- Records destruction: Ensure that records are destroyed after the required retention period ends

For a detailed introduction of records management, the architecture, and the complete offerings, see 6.2, “IBM Enterprise Records” on page 137 and refer to the first chapter of *Understanding IBM FileNet Records Manager*, SG24-7623.

7.8.2 Discovery

In addition to records management, content management solutions can also deliver facilities for finding, categorizing, and delivering content based for legal actions. For these needs, the IBM FileNet P8 Platform offers integration with the IBM eDiscovery Analyzer and eDiscovery Manager. These products provide assistance with legal discovery by enabling users to search and export documents. IBM eDiscovery Analyzer searches email, attachments, and other documents that have been collected into a case by IBM eDiscovery Manager

For more information, including architecture and UI, see section 6.5, “eDiscovery Manager and eDiscovery Analyzer” on page 157.

7.9 Monitoring

After a system is deployed, it is the responsibility of the system administrators to ensure the health and performance of the system to meet agreed upon service levels. As such, the IBM FileNet P8 Platform does include some out-of-the-box functionality for monitoring the health of the system. This is the IBM System Dashboard for Enterprise Content Management. See “IBM System Dashboard for Enterprise Content Management” on page 55 for more details.

Monitoring business activity

Business activity monitoring addresses the supervision processes in progress on the Process Engine. IBM has two offerings to use for this purpose in conjunction with the IBM FileNet P8 Platform:

- IBM Cognos Real Time Monitor
- IBM WebSphere Business Monitor (WBM)

They both serve the same need by presenting live information from the BPM back end system and from other data sources, in a real-time fashion. The data is gathered from the configured sources, aggregated to represent key information, such as service level agreements (SLA) and typically other key performance indicators (KPI) and thresholds. To represent this information to the business monitoring dashboards, which represent the information by gauges and other intuitive graphical elements, are typically used.

8

Security

Each IBM FileNet P8 module has its own functionality, but they are all built on top of the IBM FileNet P8 Platform with Content Engine, Process Engine, and Workplace or Workplace XT, which are described in earlier chapters. The support for security around authentication and access control of processes and content of these products is provided by the core platform. This chapter describes the security issues to consider in an enterprise environment, how IBM FileNet P8 addresses them, and how to manage security effectively in an IBM FileNet P8 environment.

This chapter covers the following topics:

8.1 Authentication and authorization

This section describes how the IBM FileNet P8 Platform interacts with other enterprise systems to perform authentication. The various types of objects, services, and operations that can be protected using access control are discussed and how authorization is carried out for those elements.

8.1.1 Terminology

It is important to differentiate between authentication and authorization.

Authentication deals with asserting the unique identity of a user or a service, based on one or more credentials the user must provide, for example, a personalized secret password, an individual keycard, or a fingerprint. Users and services are instances of a more general concept: the *security principal*. A security principal is an entity that can be authenticated by a computer system or network. Authentication is the process of validating and confirming the identity of such an entity.

Authorization is about defining and regulating which permissions a given user has for a given object, which might be the permission to add a text annotation to a specific document in an ECM system, for example.

Authorization allows and also denies access rights. The following terms are used in relation with authorization:

- Explicit access right: The given access right (allow and deny) will explicitly be named in the access control list (ACL) of a given object.
- Implicit access right: All access rights that are not explicitly set will be implicit denied.
- Direct access right: The access right will be added straight on the ACL of a given object (either by default instance security or manually editing the ACL of a given object)
- Inherited access right: The access right will be related by reference to another object's ACL

8.1.2 Authentication in IBM FileNet P8

IBM FileNet P8 must use a directory server to authenticate against and be able to look up user group membership information using Lightweight Directory Access Protocol (LDAP). Starting with version 4.0, IBM FileNet P8 uses the application server's built-in support for authentication to a central directory server.

IBM FileNet P8 currently supports the following directory servers:

- Microsoft Active Directory 2003, 2008 SP1+, 2008R2
- Microsoft 2003 Active Directory Application Mode (ADAM)
- Microsoft 2008 SP1+, 2008R2 Active Directory Lightweight Directory Service (AD LDS)
- IBM Tivoli Directory Server 6.0, 6.1, 6.2, 6.3
- Sun Java Directory Server 5.2 SP3+, 6.3+; 7.x
- Novell eDirectory 8.7.3, 8.8.x
- CA Directory
- Oracle Internet Directory 10g (10.1.4.0.1+), 11g

The current list of the supported authentication providers is in the IBM FileNet P8 Hardware and Software Support Guide, which is available from the IBM web site at:

http://www.ibm.com/support/docview.wss?rs=3278&uid=swg27013654

Also, see the Support Matrix of each directory server for lookup of supported directory features.

FileNet P8 supports two ways of authentication:

- Java Authentication and Authorization Services (JAAS)
- Web Services Security standard (WS-Security)

The following descriptions summarize the various supported default methods of authentication within FileNet environments. For detailed information and possibilities to enhance the standard feasibility, see the authentication area within the security section of the information center:

http://publib.boulder.ibm.com/infocenter/p8docs/v5r0m0/index.jsp?topic=/com.ibm.p8.security.doc/p8psn036.htm

Java Authentication and Authorization Services

The FileNet Workplace and the Workplace XT use JAAS authentication in container-managed authentication mode. In this mode, the deployment descriptor for the application specifies the security constraints required to access application pages. Perimeter authentication is used for standard single sign-on (SSO) implementations of the FileNet modules. IBM Tivoli Access Manager and CA SiteMinder are currently qualified for FileNet 5.0.

Some limitations around SSO have to be considered:

- JAAS-based SSO integration is available only to J2EE-based clients (clients who can perform a JAAS login). This type of client includes most of those that are browser-based and work with a J2EE-based presentation-tier server.
- JAAS-based SSO integration generally does not extend to .NET-based clients or pure web-services-based clients.
- JAAS-based SSO integration is generally available only if the client's J2EE environment is supplied by the same application-server vendor hosting the Content Engine application.
- The configuration of SSO solutions generally requires a high level of product-specific expertise. Before installing a FileNet P8 solution, customers must work with their SSO and J2EE application server vendors to correctly configure the single sign-on environment. The industry-leading SSO products are designed to be highly flexible, supporting a wide range of credential types and configuration options. FileNet can only test a limited number of the infinite possible combinations of SSO vendor product configurations and application servers.
- In all cases, the third-party SSO product must be configured to work with the same underlying directory service (for example, Microsoft Active Directory and Novell e-Directory) used to resolve user and group credentials within FileNet P8.

After a caller is authenticated by a J2EE Servlet container, if the Servlet subsequently calls an EJB, the Servlet Container is required to propagate the caller's identity (JAAS Subject) to the EJB. Figure 8-1 on page 195 shows the container-managed authentication case as an example using forms-based authentication to authenticate the caller against an Active Directory service.

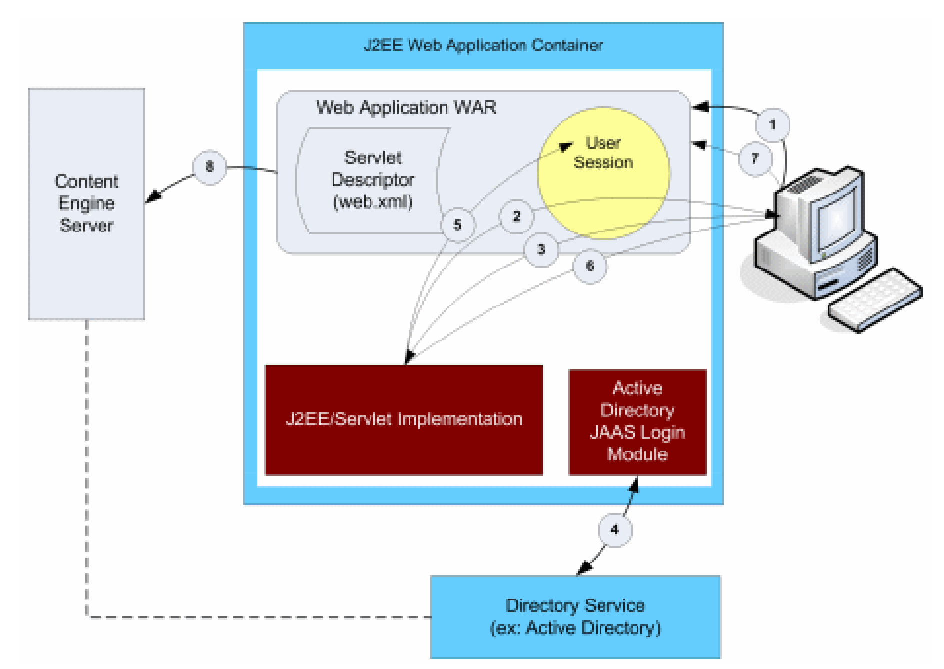

J2EE Web Application Container
Web Application WAR
Servlet Descriptor (web.xml)
User Session
Content Engine Server
J2EE/Servlet Implementation
Active Directory JAAS Login Module
Directory Service (ex: Active Directory)
1
2
3
4
5
6
7
8

Figure 8-1 JAAS container managed authentication

The step-by-step walkthrough for the authentication scenario in Figure 8-1 is:

1. A user attempts to access a Servlet-based application.
2. The J2EE application server redirects the user to a page that requires credentials.
3. The user enters credentials and submits them to the server.
4. The J2EE server validates the user's credentials through JAAS.
5. The J2EE server creates JAAS principals and subject objects using the Active Directory JAAS Login Module and places them in the user's session.
6. The J2EE server redirects the user back to the application page that was originally requested.
7. The Servlet container looks for a user principal available on the incoming request.
8. After invoked, the servlet makes a call to the P8 Content Engine server, and the user's JAAS subject is propagated to the EJB container in which the Content Engine resides.

Because of this approach, the only piece of information that the Content Engine uses from the JAAS context is the identifier of the user. Regardless of the JAAS module used, this identifier must be consumable by the Content Engine's configured LDAP user and group lookup filters. Typically this is accomplished by using the LDAP common name field, for example, the user name on the system.

The Content Engine does not use JAAS to get any information about group membership, roles, or permissions on objects. All group membership is looked up through LDAP by the Content Engine. All permissions are stored in the relevant Content Engine or Process Engine databases for that particular user's unique identifier. For an IBM FileNet P8 system that is linked to Active Directory, for example, the unique identifier is the user's SID, which means that the LDAP directory schema does not need any changes to it to support an IBM FileNet P8 solution. The only change that is required is creating users and groups to manage security of the system.

Lookup: In IBM FileNet P8 4.0 and above, all authentication and group membership lookup is delegated to the Content Engine. The Process Engine no longer performs its own lookups directly to the Directory Server, which simplifies the configuration of Authentication and Single Sign-On.

Although the JAAS standard is designed to abstract authentication and authorization, it occurs at the cost of mapping constructs into a tight JAAS model. The Content Engine has fine-grained support for access control, so it cannot use the Java security access control model. Many JAAS providers also do not support the more advanced security features of the most common directory servers. An example of this is Microsoft Active Directory's support for multiple forests with multiple domains. As such, IBM FileNet P8 performs its own group membership lookups and maintains its own access control lists internally.

This greater flexibility requires the developers to write code specifically to talk to each directory server. As a result, IBM FileNet P8 supports a specific subset of the most prevalent directory servers that are available on the market.

WS-Security

Two WS-Security profiles, the user name Token profile and the Kerberos profile, are supported by FileNet P8. Support for additional standard profiles and nonstandard WS-Security compliant approaches can be integrated with FileNet P8 Web services, using the FileNet P8 Web Service Extensible Authentication Framework. FileNet P8 does not provide any explicit support for SAML tokens. However, the Web Services Extensible Authentication Framework can be used to build support for authenticating through SAML-based credentials within FileNet P8 applications.

Figure 8-2 illustrates the WS-Security authentication.

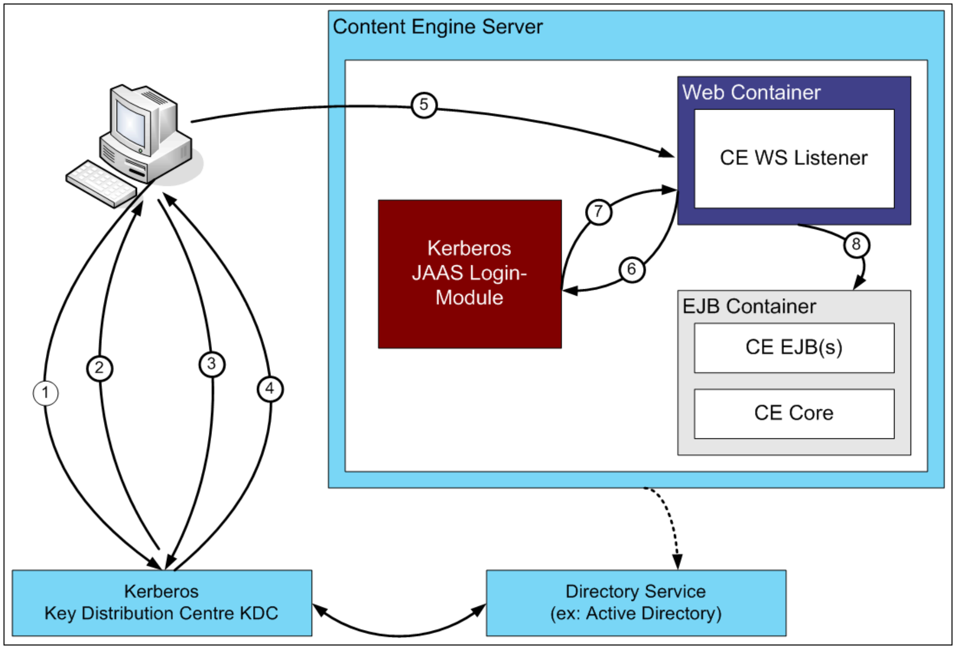

Content Engine Server
Web Container
CE WS Listener
Kerberos JAAS Login-Module
EJB Container
CE EJB(s)
CE Core
Kerberos Key Distribution Centre KDC
Directory Service (ex: Active Directory)
1
2
3
4
5
6
7
8

Figure 8-2 WS-Security authentication

The steps for the authentication scenario in Figure 8-2 are:

1. For authentication purposes, the client logs into the Windows domain. The operating system collects the user's name and password and sends them to a KDC.
2. The KDC works with a directory service to validate the user's credentials and returns a ticket-granting ticket (TGT) to the caller.
3. Because the client wants to access a particular server, a ticket-granting exchange must occur. The client sends a second request to the KDC, specifying the service that it wants to access. (If the client was previously granted a service ticket that has not yet expired, then steps 3 and 4 are skipped).
4. The KDC issues a service ticket that grants the caller access to that service.
5. The client sends its request with the service ticket encoded in a WS-Security Kerberos token to the server.
6. The Web service listener on the Content Engine server performs a JAAS login using credentials supplied in the incoming WS-Security header.

7. The JAAS Login Module validates the client's ticket. No round-trip to the KDC is necessary. The server can use its own key (obtained from the KDC at startup time) to validate service tickets. The Login Module produces a valid JAAS subject based on the successful validation of the ticket. This JAAS subject is returned to the Content Engine Web service listener.
8. The Content Engine Web service listener passes the call along (with valid JAAS subject) to the Content Engine EJB layer.

For limitations concerning Microsoft’s Kerberos implementation, refer to the security information provided in the IBM FileNet P8 Version 5.0 information center.

Process Engine authentication

The Process Engine provides two different algorithms for authentication:

- Authenticating the Process Engine Java API client using Content Engine EJB transport and
- authenticating the Process Engine Java API client using the Content Engine Web Service transport.

Process Engine Java API client using Content Engine EJB transport

Figure 8-3 on page 199 shows the authentication process for the Process Engine Java API client using Content Engine EJB transport.

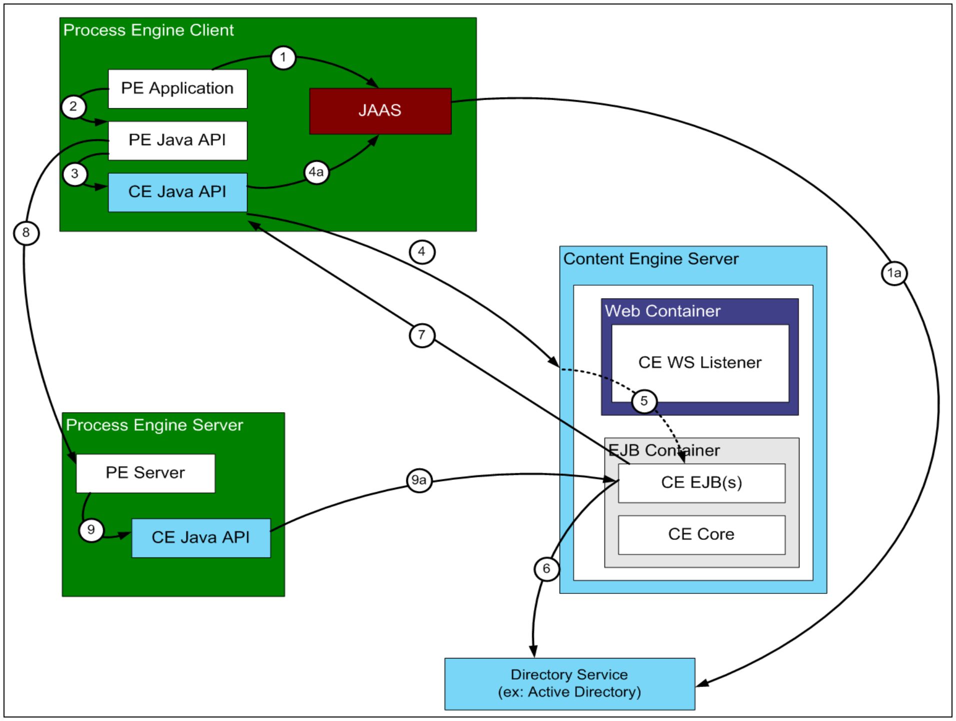


Figure 8-3 Process Engine Java API client using Content Engine EJB transport

Figure 8-3 shows the two-level authentication process, where the client first authenticates with JAAS. The client then authenticates with Content Engine, with the application server validating the JAAS credentials by invoking one of the configured Login Modules to confirm that the client's credentials are of a type accepted by this server and that they are valid. This process establishes the caller's identity and provides access to FileNet P8.

In detail, the following steps are executed:

1. The Process Engine client performs a JAAS login using the Login Module configured in the client environment. This login probably occurs before any interaction occurs with the Content Engine or Process Engine APIs. It can happen automatically as a result of a J2EE configuration, or the client can invoke the Login Module programmatically.

   Depending on what Login Module is in use, the Login Module can make a call to the enterprise directory service, possibly through a proxy or some intermediate SSO solution.

2. The Process Engine Application makes a call to the Process Engine Java API.

3. The Process Engine Java API sees that the client is not authenticated to FileNet P8 yet, and therefore makes a call to the Content Engine Java API to obtain a FileNet P8 identity token.
4. The Content Engine Java API makes a call to the server. At the Content Engine server, the call arrives at the J2EE application server's EJB container with the caller's JAAS Subject. The application server examines its security policy configuration and how the Content Engine EJB is configured to determine what security policy is in place. The application server determines whether the JAAS Subject associated with the incoming request matches one of the authentication providers that are configured and enabled for use with the EJB. If it does, the application server makes a call to this authentication provider which performs any necessary checks on the JAAS Subject (this might involve contacting a directory service or SSO provider to validate the Subject). If any part of this application server authentication process fails, an appropriate error is returned to the caller (this occurs before any logic within the Content Engine EJB is invoked).
5. If the incoming JAAS Subject is validated by the application server, the call is passed through to the Content Engine EJB, and the JAAS Subject is available to the Content Engine EJB.
6. Within the Content Engine EJB, the Principal name is extracted from the JAAS subject, and searched for in the Content Engine user cache. If not found, Content Engine calls the enterprise directory service to expand the caller's identity information.
7. The EJB processes the incoming request. In this case, a FileNet P8 identity token is created and returned to the caller (the Process Engine Java API).
8. If all the previously mentioned steps are successful, the Process Engine Java API now has a FileNet P8 identity token, obtained from a Content Engine server. The Process Engine Java API makes a call to the Process Engine server, passing the identity token as a parameter. The Process Engine server examines the identity token, and the signature on it is validated. If the signature is valid, the Process Engine server trusts that the token and the user identified by the token are valid.
9. This optional step is unrelated to the authentication process, but occurs in many cases. In this step, Process Engine wants to retrieve the detailed user information (full name, DN, email address, group memberships, and so on). It does so by calling Content Engine (9a) to retrieve the requested user and group objects.

Process Engine Java API client using the Content Engine Web Service transport

Figure 8-4 on page 201 shows the Process Engine Web service listener, which is always co-resident with the Content Engine server.

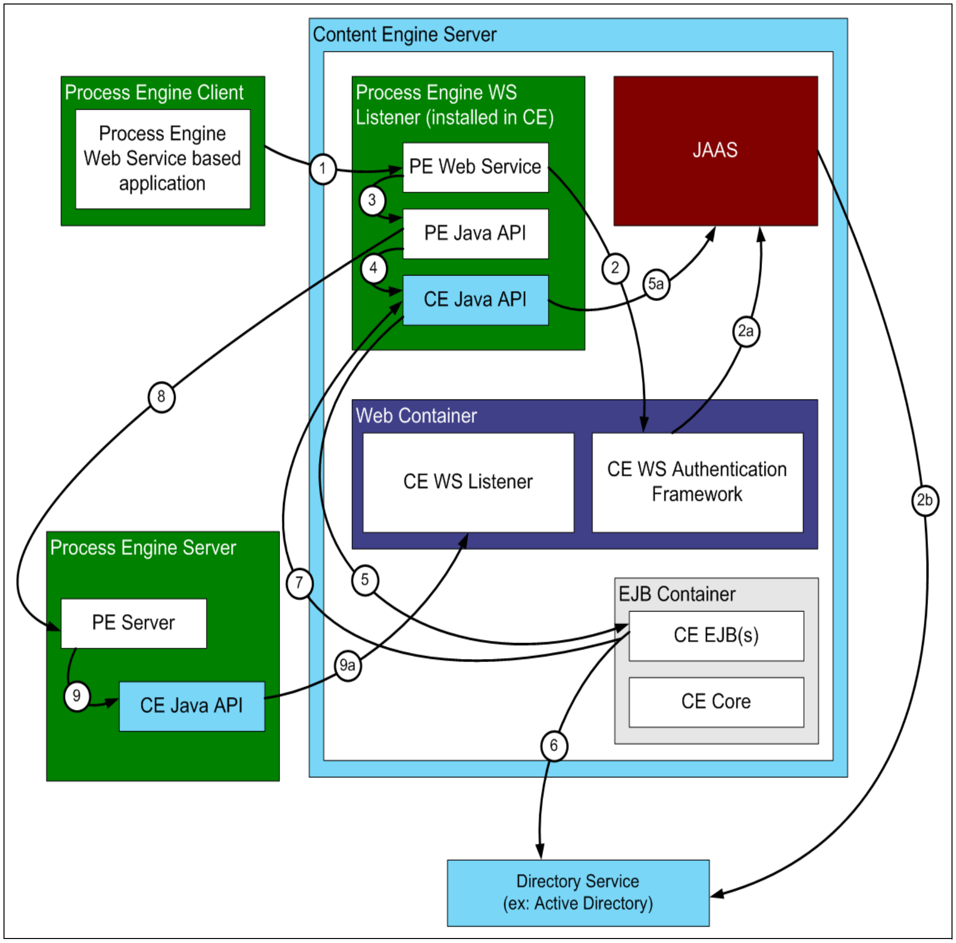


Figure 8-4 Content Engine responding to Process Engine Web service-based call

The following steps occur as Content Engine responds to a Process Engine Web service-based application call:

1. The Process Engine client sends a web service request to the Process Engine Web service listener.
2. The Process Engine Web service listener obtains the JAAS subject by using the Content Engine Web Services Authentication Framework, which expects to find the user name and password in the SOAP Header based on the WS-Security user name Token profile.
3. The Process Engine Application makes a call to the Process Engine Java API.

4. The Process Engine Java API sees that the client is not authenticated to FileNet P8 yet, and therefore makes a call to the Content Engine Java API to obtain a FileNet P8 identity token.
5. The Content Engine Java API makes a call to the Content Engine EJB home interface with the JAAS Subject obtained in step 2.
6. Within the Content Engine EJB, the principal name is extracted from the JAAS subject and searched for in the Content Engine user cache. If not found, Content Engine calls the enterprise directory service to expand the caller's identity information.
7. The EJB processes the incoming request. In this case, a FileNet P8 identity token is created and returned to the caller (the Process Engine Java API).
8. If all the previously mentioned steps are successful, the Process Engine Java API now has a FileNet P8 identity token, obtained from a Content Engine server. The Process Engine Java API makes a call to the Process Engine server, passing the identity token as a parameter. The Process Engine server examines the identity token, and the signature on it is validated. If the signature is valid, the Process Engine server trusts that the token and the user identified by the token are valid.
9. This optional step is unrelated to the authentication process, but occurs in many cases. In this step, Process Engine wants to retrieve the detailed user information (full name, DN, email address, group memberships, and so on). It does so by calling Content Engine to retrieve the requested user and group objects.

Supported credentials: The only credential type supported for this scenario are user name and password credentials. The Process Engine Web service does not support the Web Services Extensible Authentication Framework (WS-EAF).

ECM Widget integration

IBM ECM Widgets are only available for IBM WebSphere Application Server-based deployments. The user will login through Business Space or Lotus Mashups (depending on which container was installed). When the widgets are set up, Business Space (or Mashups), Workplace XT, and the ECM widgets war are put in the same WebSphere Application Server profile and configured for SSO so that they can share the login session through WebSphere Application Server LTPS. The Process Engine representational state transfer (REST) services and an internal ECM widgets-provided private REST service that communicates with Content Engine are installed as servlets in Workplace XT.

Figure 8-5 on page 203 shows the Widget integration.

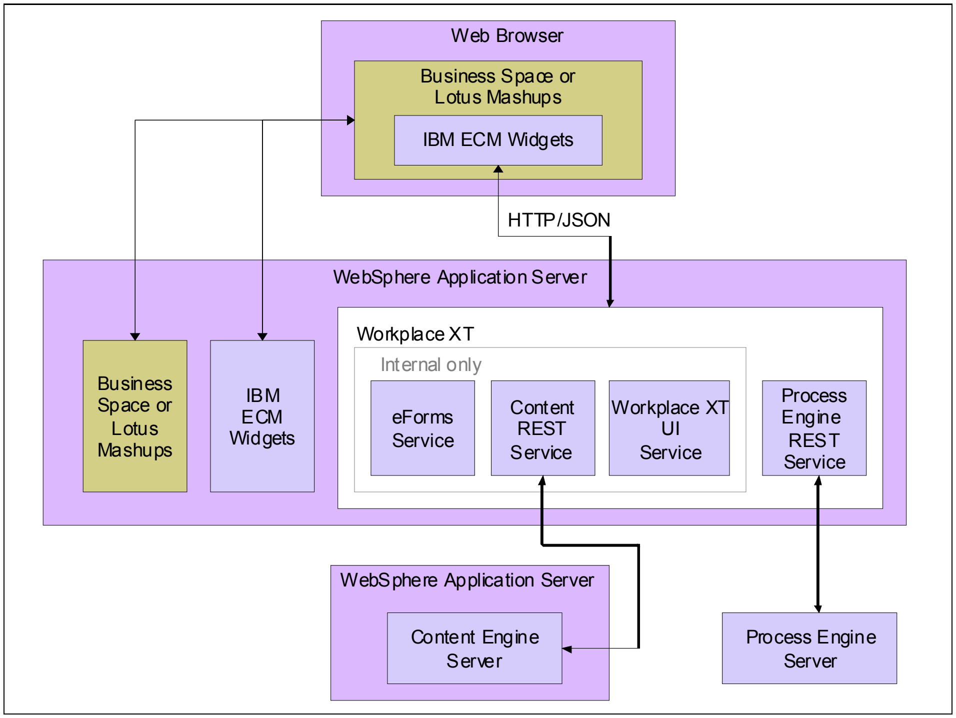


Figure 8-5 Widget integration in IBM FileNet P8

For additional information about access roles for Widgets and the handling with IBM WebSphere Business Spaces see:

`http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp?topic=/com.ibm.websphere.wbpm.bspace.config.620.doc/doc/tcfg_bsp_app_admin_security.html`

8.1.3 Separated authorization

While authentication and authorization are fundamentally different in nature, they often are addressed with the same types of tool and with exactly the same tool instances, namely the same enterprise directory (respectively LDAP provider). This directory contains user and credential information for authentication and group and group membership information for authorization.

Authentication can be used commonly for all IT applications of an organization because the users are the same. The leading advantage having a common authentication is the ability—as long the application will support this kind of authentication—of a Single Sign-On (SSO).

For various reasons it might be necessary to separate authentication and authorization, which can be assigned to individual applications having their own directories for authorization purpose.

Reasons for separated authorization are, for example:

- The used LDAP already has a huge amount of groups and group memberships so that the security information exceeds the configurable Kerberos limits.
- Limited or forbidden access to the default LDAP.
- Organizational reasons.

IBM FileNet P8 allows for this separation following the concept of an application server channel for authentication and a Content Engine channel for authorization. The first is configured as part of application server security, while the latter is configured within the application (that is, the LDAP provider for the P8 Content Engine application).

In real-life projects, the separation of those two can be implemented with the application server channel pointing to a Kerberos ticket validation authority and the Content Engine channel pointing to an application specific Active Directory Lightweight Directory Services LDAP provider holding the group membership information to represent the permission groups and roles used in the IBM FileNet P8 environment.

To enable the separation, there must be a custom application established to ensure that the following requirements are fulfilled:

- Every user-account must have its equivalent in the authentication directory as well as in the authorization directory. Changes (for example, disabling of accounts) must be synchronized.
- Group memberships must be related in the authorization directory.

Note: Usually not only the application that technically fulfills these requirements (How will that be managed) must be individually created, but the definition of the user/group relationships must be automated (Why will the given relationship be created) by custom applications as well.

As of IBM FileNet P8 5.0 a major change in the use of the LDAP was implemented. Prior to release 5.0, the Security Identifier (SID) must have been related to the GUID of the used LDAP. As of Release 5.0, the LDAP attribute(s) used for the SID can now be chosen during the initial set up of Content Engine Directory Configurations:

- When set, values from the configured SID attribute are used for SIDs in Content Engine permissions rather than those from the default SID attribute.
- Can use the Distinguished Name (DN) as the SID attribute even though the DN is not an attribute in LDAP servers.
- Can set one SID attribute for users and another for groups.
- The configurable SID is set using IBM FileNet Enterprise Manager (FEM) or using the new Content Engine API properties: UserUniqueIDAttribute and GroupUniqueIDAttribute (which can be set only on create).

Important: It must be ensured that the used login attribute value for authentication is unique. If more than one security user or group was found in the configured realms that has the given attribute value, a *SECURITY_TOO_MANY_MATCHES* error is displayed. Authentication is not possible for the given account. Content Engine requires this given attribute value to be unique across the configured realms.

8.1.4 Single Sign-On

The term Single Sign-On, or SSO, is often misused and generally implies that there is no manual sign required in several scenarios. For clarity, the following definitions of methods of sign on are used:

- Traditional authentication

  User name and password-based sign on made manually by users.

- Single Sign-On (SSO)

  This is when a client authenticates to the first SSO-protected service that they use and does not have to log into any other. In other words, users sign in only one time. Before a user request is handled, an SSO provider is called that intercepts the request and authenticates the user. The SSO provider can either be one of the currently qualified end-to-end solutions (IBM Tivoli Access Manager and CA SiteMinder) or an SSO vendor has provided Java Authentication and Authorization Service (JAAS) Login Modules that work with a given application server. This user session is then asserted as valid to the application, so the user is not again asked to supply their user name or password, which is the process when using SSO software, such as Tivoli Access Manager with WebSeal.

  Typically, the same gate keeper SSO software is used across multiple Web interfaces and signing into one of these means the user does not need to sign into the others. Web application SSO protection systems, such as secure reverse proxy software like WebSeal, provide their own login page instead of using the underlying services' login page.

  A common mechanism for SSO is Kerberos, which is the underlying technology that is used in Windows Integrated Login, which uses the same method when passing Kerberos tokens to applications. For Windows Integrated Login, the user's machine retrieves the Kerberos token from the Directory Server during initial login and passes it directly to the application server to validate. In this situation, the application server itself acts as the intercepting third party. Instead of the application server presenting a login prompt, it asks for a token but does not prevent access to the underlying application if it is not present.

- Token passing

  Token passing occurs between two web-based applications that share a trusted relationship. Within Workplace, for example, a user might click a hyperlink that opens an Enterprise Records application. This URL also holds the user token so that Enterprise Records does not ask the user to log in again.

Notice that Single Sign-On requires third-party software to validate the user, which allows developers to abstract out any authentication code from the underlying application. Organizations can implement the same security access restrictions across all compliant applications, which is exactly what the Content Engine supports through JAAS.

The method used to assert credentials to a service where access is controlled by gate keeping SSO software can be one of two types. Under the first approach, some products can be configured to fake the login to the application, which allows the SSO software to look up the user name and password and passes that into the login box of the Web application being used. This method logs in the user in the background and then performs the user's request. The protected application has no idea that the client has not typed in an actual user name or password. This method is useful when adding SSO support to older applications that do not check for JAAS contexts.

The second method, and the method that Workplace XT uses, is to assert a JAAS context to the application server and have the application check for a valid context before forcing a manual login. This context can then be passed through the application stack to other JAAS-enabled software with which it communicates, such as the IBM FileNet P8 Content Engine and Process Engine. Figure 8-6 on page 207 shows a typical process for how Single Sign-On works with a third-party SSO provider.

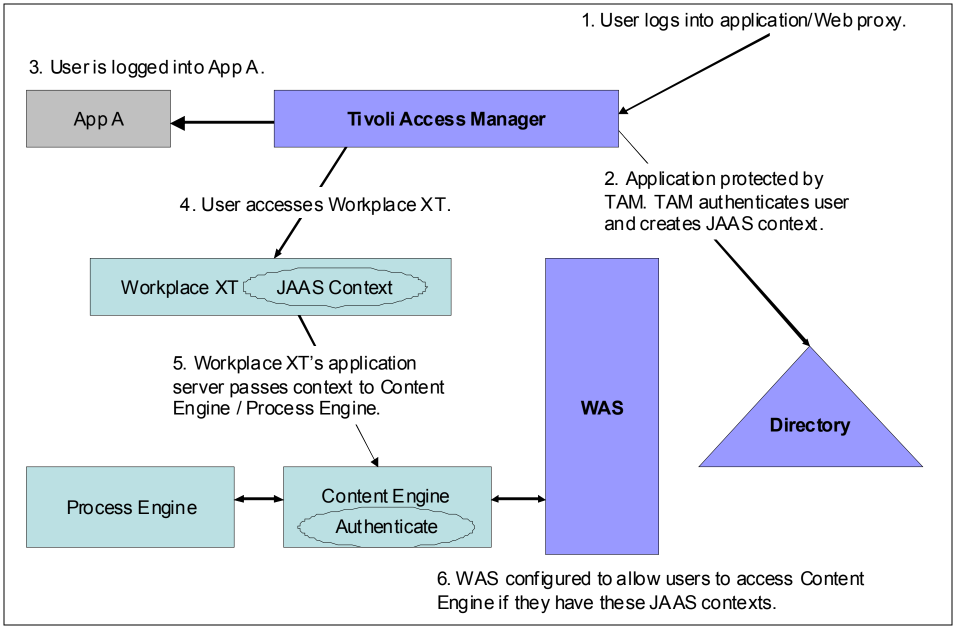

1. User logs into application/Web proxy.
3. User is logged into App A.
App A
Tivoli Access Manager
2. Application protected by TAM. TAM authenticates user and creates JAAS context.
4. User accesses Workplace XT.
Workplace XT
JAAS Context
WAS
Directory
5. Workplace XT's application server passes context to Content Engine / Process Engine.
Process Engine
Content Engine
Authenticate
6. WAS configured to allow users to access Content Engine if they have these JAAS contexts.

Figure 8-6 Typical Single sign-on (SSO) authentication behavior

Some application servers support token or SSO-based authentication themselves. An example of this is the WebSphere 6.1 support of the Simple and Protected GSSAPI Negotiation Mechanism (SPNEGO) client handshake method and in particular the Kerberos token that can be retrieved using that method. This requirement is common, especially in organizations that widely use Windows Integrated Login for their employee-accessed applications.

Note: The Content Engine can be used with any JAAS context configured in the application server. For latest support information, refer to the hard- and software requirement information provided in *Hardware and Software Requirements Version 5.0* chapter *Directory servers* at:

https://www-304.ibm.com/support/docview.wss?uid=swg27013654

There is a wide range of SSO authentication options that are available today. They are typically tightly linked to the application server and directory server being used. For a list of specific IBM FileNet P8 versions that are supported for each configuration, check the latest IBM FileNet P8 Hardware and Software Support Guide, which is available on the IBM web site.

For more information see:

- IBM Redbooks publication

  Single Sign-On Solutions for IBM FileNet P8 Using IBM Tivoli and WebSphere Security, SG24-7675
- IBM Information Center IBM FileNet P8 section security subsection SSO:

  http://publib.boulder.ibm.com/infocenter/p8docs/v5r0m0/index.jsp?topic=/com.ibm.p8.security.doc/p8psn041.htm
- IBM Information Center IBM WebSphere Application Server section security subsection SSO:
  http://publib.boulder.ibm.com/infocentre/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.express.doc/info/exp/ae/csec_sso.html

8.2 Securing core P8 components and resources

The IBM FileNet P8 modules offer much functionality that can be secured independently. In this section (8.2, “Securing core P8 components and resources” on page 208), a high-level summary of each component and the circumstances under which it makes sense to secure them are described. These areas are discussed in more detail in the following sections (as of chapter 8.3, “Access to information” on page 214).

Two different types of security must be discussed: securing documents and metadata from misuse and unauthorized access and the security that is needed for administrative purposes, such as installing and maintaining the system with all its facets.

8.2.1 Content Engine

The Content Engine has the richest set of authorization access controls of any part of the IBM FileNet P8 Platform, which is necessary to satisfy a range of use case scenarios. Thinking about security for the Content Engine is naturally thinking about access to document content and metadata but, in reality, access control goes far beyond that.

The Content Engine supports a wide range of permissions some of which are applicable only to certain types of objects, for example, the *Major Versioning* or *View Content* permissions are only applicable to instances of the document class and its subclasses. The *View All Properties* permission is applicable to documents, custom objects, folders, and many more objects in the Content

Engine. The most commonly used permissions are described in chapter 8.3.1, “Security of Content Engine objects” on page 215.

At the heart of all the methods of assigning permissions in the Content Engine is the concept of *Access Control Entries* (ACEs). An ACE entry links a permission to a user or group in the directory server. ACEs are contained within an ACL in IBM FileNet P8 terminology that is attached to a Content Engine object.

ACLs are assigned to an object and are not classed as objects in their own right, which means that one cannot assign the same ACL to multiple objects. The Content Engine does, however, check whether newly created ACLs are identical to ones that already exist within the system. If the new ACL is the same as one that already exists, the Content Engine only stores one copy. This function is transparent to applications that are built on top of the Content Engine and allows for efficient caching of permissions.

The security objects that the Content Engine supports are Marking Sets, Security Policies, Document life cycle Policies, Dynamic Security Inheritance Objects, and Default Instance Security descriptors, and each of these are described in subsequent sections. An instance of a Content Engine object, such as a Document, maintains its own list of instance security, independent of the previously mentioned security mechanisms. This is populated when an instance is created by the Default Instance Security settings of the document class or specified explicitly by the application that added the document.

Workplace XT, for example, has the concept of Document Entry Templates. There can be many types of *Entry Templates* for a single class of document that the user chooses. These templates can set default values for properties, set a default filing location, and specify security ACEs. They are application-level constructs that are independent of built-in descriptors that are used within the Content Engine.

Entry Templates are useful in certain circumstances, for example, if a clerk enters a document to the Content Engine it is reasonable that the ACLs are defined automatically without any user intervention.

An interesting capability of the highly granular security model in the Content Engine is that it is possible to lock system administrators out from viewing the content using marking sets but still allow them to effectively maintain the system. This feature is especially useful in organizations, such as those in Financial Services, Healthcare, or Defense, where the need to secure information separately from the infrastructure might be a requirement. Because it is possible to lock such employees out of accessing Content Engine information, it can be ensured that stringent security policies cannot be circumvented through administrative or maintenance access.

8.2.2 Process Engine

The Process Engine has a concept of *User Queues*, *Work Queues*, and *Component Queues*. A user queue restricts access to work items in that queue to only the specific user that is assigned in the workflow definition. A Work Queue is accessible by multiple users and groups who all share a similar role within a workflow.

The following list contains IBM FileNet P8 terminology that is used in the context of workflows:

- Queue: A list of active work items grouped logically. A queue can contain several types of activities to be worked on by groups or users with allowed access (based of the particular security definitions) to that queue.
- Step: An item of work to be completed by either a user or a background system.
- Process Map: Also called a workflow definition. An executable definition of steps, routing conditions, and fields to be carried out by the Process Engine.
- Process: Also called a workflow. A running instance of a Process Map with its own data values and processing history.
- Isolated Region: An area in the Process Engine database that contains a group of processing queues, their configurations, related transferred workflow definitions and dependant work items and their related processing entries.
- Roster: A workflow roster is a database structure that stores information about a group of workflows in an isolated region. An isolated region can contain more than one roster.

A Component Queue is used to interact with instances of a Java class to perform some system task, which can be to file a document in a particular location set, fetch a document's properties, or update document security. Because this component is a system process, it is possible to specify a particular JAAS stanza (configuration) to use and the user name and password to pass to the configured login modules, for example, login modules exist for user name and password authentication to the appropriate Process Engine, Content Engine's Java APIs.

It is also possible to construct a component that asserts its own custom JAAS context using standard Java code for cases where it might be needed to interact with other third-party systems. This action is dependent on how the component is coded by the developer and independent of IBM FileNet P8.

Certain IBM FileNet P8 add-on products extend the security that is available in the Work and User Queues to restrict the list of work items that are displayed in the interface. Business Process Framework (BPF), for example, can be configured to specify an inbasket configuration that restricts a particular queue to a specified role, which can be further restricted by specifying that only a single step type is shown in this inbasket, or by constructing a queue filter to show only certain items based on the value of a process property. A good example of this is when there are junior and senior staff members with differing access authority. A Junior Approver queue filter can be created to restrict the list of pending credit approvals to those credit requests that are less than $10,000. This feature is also a useful way to limit the number of work items being displayed for a particular queue because it is possible to create more than one inbasket that lists a subset of the work items that are available to that user in the queue.

8.2.3 IBM FileNet P8 user interface access roles

The IBM FileNet P8 user interfaces (UI) extend the concept of user and group-based access control by restricting who can see and use certain functionality that is available in the interface to particular users and groups.

A Workplace XT role, for example, is simply a list of users and groups. An authors role can be created to allow only certain people to see the “Add document” or “Check in” user interface actions. This approach covers all actions that are available from the right-click document and folder menus and on the property pages for objects. Roles restrict who can see the main pages, such as Browse, My Workplace, Tasks, and Search.

Roles are commonly used to restrict who can see advanced authoring tools, such as the Search Template Designer or Process Designer. There is even a special role, called P8BPMProcessDesignerEx (is introduced when installing and configuring IBM FileNet Connector for Microsoft Visio), that lists who can see the Microsoft Office Visio diagram import option within the Process Designer applet.

Business Process Framework has its own concept of roles. BPF can be configured to use LDAP group integration where the role name is the same as a group in the directory server. Another alternative is to link a BPF role directly to an existing Workplace role with the same name, which is useful if your users access both user interfaces because it is only necessary to specify the members of each role one time rather than for both products. Another benefit is that users of BPF can have access to their queues removed from the Tasks interface of Workplace, which forces them to go through BPF to work on their tasks.

My Workplace acts as a mini portal where custom pages can be configured to show certain folders, search results, or actions. These pages can be restricted down to certain Workplace access roles. This restriction is particularly useful where an organization wants to deploy a configurable portal solution for ECM and BPM but does not want the overhead or cost that is associated with a full-blown enterprise portal solution.

8.2.4 Security aspects for installation and maintenance

Besides the security configuration for accessing the FileNet P8 environment for business purposes, the original intention of introducing an ECM system is that there are administrative security requirements for installing and maintaining the different modules.

As the general installation conditions vary from the dedicated security requirements of the contemplated environment. Usually production environments must fulfill the highest requirements within a dedicated company, and it has proven its worth to implement at least one test environment with a comparable security configuration.

Security requirements for installing IBM FileNet P8

There are various systems touched, when installing the core modules of the IBM FileNet P8 modules. This chapter discusses security requirements for security configuration of the communication between the systems because there are far too many different combinations in use. At this point, usually the following requirements must be considered:

- Protocols for communication between the modules under observance of the relation of the modules and the server—physical or virtual— the software is installed on. See also Figure 2-1 on page 20 for an extract of used protocols.
- Information concerning network addressing, such as MAC-addresses, IP-addresses, or ports.
- The use of ActiveX and JavaScript.

Also consider that firewalls can be required between client (applications) and servers as well as within the server environment. Additional purposes for administration, backup, recovery, and monitoring also must be regarded.

Administrative accounts: To prevent dependencies on single points of failure (loss of passwords) and to use dedicated and individual administrative accounts, configure groups instead of dedicated user accounts whenever applicable.

See Table 10-2 on page 361 in Appendix A, “User accounts” on page 359, which shows an overview of security requirements of user accounts related to a core IBM FileNet P8 environment (including database accounts) that are needed for installation and initial maintenance configuration.

For the current overview and a detailed description, see the security section in the Information Center subsection Users and groups required by FileNet:

http://publib.boulder.ibm.com/infocenter/p8docs/v5r0m0/index.jsp?topic=/com.ibm.p8.security.doc/p8psu000.htm

Requirements for password changes

Table 8-1 lists the technical users for permanent configuration that can be involved when passwords within the core modules must be changed. For additional modules further changes might be necessary.

Table 8-1 Overview of the technical accounts

Function	Account name	Where to change and remarks
Content Engine service user (LDAP read access)	*ce_service_user* (can also be used as *ce_bootstrap_admin)*	Logon to FEM as *gcd_admin* → Rootnode (Properties) → Tab: *Directory Configuration* CEMPBoot.properties file
Content Engine system user (bootstrap administrator)	*ce_bootstrap_admin* (can also be used as *cce_service_user)*	Logon to FEM as *gcd_admin* → Rootnode (Properties) → Tab: *Directory Configuration* CEMPBoot.properties file
G CD Administrator	*gcd_admin* (if used for component queue configuration)	Logon to PCC as *pe_region_admin* → *Region[x]* → *Component Queues* → *CE_Operations (properties)* → *Tab: Adaptor*
K2 security user	*k2_sec_user*	Logon to FEM as *gcd_admin* → Rootnode (Properties) → Tab: *Verity Domain Configuration* → select domain → *Edit*
Process Engine service user	*pe_service_user*	Logon to FEM as *gcd_admin* → *PE Region Ids* → select region → Tab: *General* Logon to PTM as *pe_region_admin* → PE node → Tab: *Security* Windows systems (autostart only) *Start* → *Control Panel* → *Administrative Tools* → *Services* → *<component> Services Manager* → Tab: *Log On* where <component> is the name of the FileNet P8 component where Process Task Manager is running

Function	Account name	Where to change and remarks
Content Engine database accounts	*ce_db_user*	IBM WAS Logon to WAS as *ce_appserver_admin* → Security → *Global security* → Area: *Authentication* → *Java Authentication and Authorization Service* → *J2C authentication data* Oracle WebLogic Logon to WL admin console as *ce_appserver_admin* Logon to *admin console* → *Services* → *JDBC* → *Data Sources* → *data source of interest* → Tab: *Configuration* → subtab: *Connection Pool* JBoss within the configuration file the username and password of the dedicated data source will be found in the xml-tag <data sources>
Process Engine database accounts	*pe_db_user*	Logon to PTM as *pe_region_admin* → PE node → Tab: *Database* → Tab: *General*
K2 operating system user	*k2_os_user*	*Windows systems* Start → Control Panel → Administrative Tools → Services → Verity K2 6.1 Administration Server service → Tab: Log On
CSS operating system user	*css_os_user*	Windows systems *Start* → *Control Panel* → *Administrative Tools* → *Services* → *Content Search Service* → Tab: *Log On*

The number of possible *user account name-technical role* combinations is quasi arbitrary and will have in real-life systems additional complex levels by additional modules and custom applications. Refer to the actual documentation for additional user account and group information, detailed configuration descriptions, and limitations.

8.3 Access to information

In this section, methods are described that are available within the Content Engine for creating access control lists. This information is available in more detail in the IBM Redbooks publication, *IBM FileNet Content Manager Implementation Best Practices and Recommendations,* SG24-7547.

It is of importance to the whole platform because all add-on products are built upon this security framework. In this section, specific references are made to where each method is used within expansion products and how overall security benefits from a platform architecture approach are showed.

Each securable object in the Content Engine maintains its own list of access control permissions, known as Access Control Entries. When several of the ACEs are attached to one object, it is known as an Access Control List. These ACLs can be specified at the document instance level to specify who can access particular documents. There are methods whereby ACLs can be reused across multiple objects, which is discussed later.

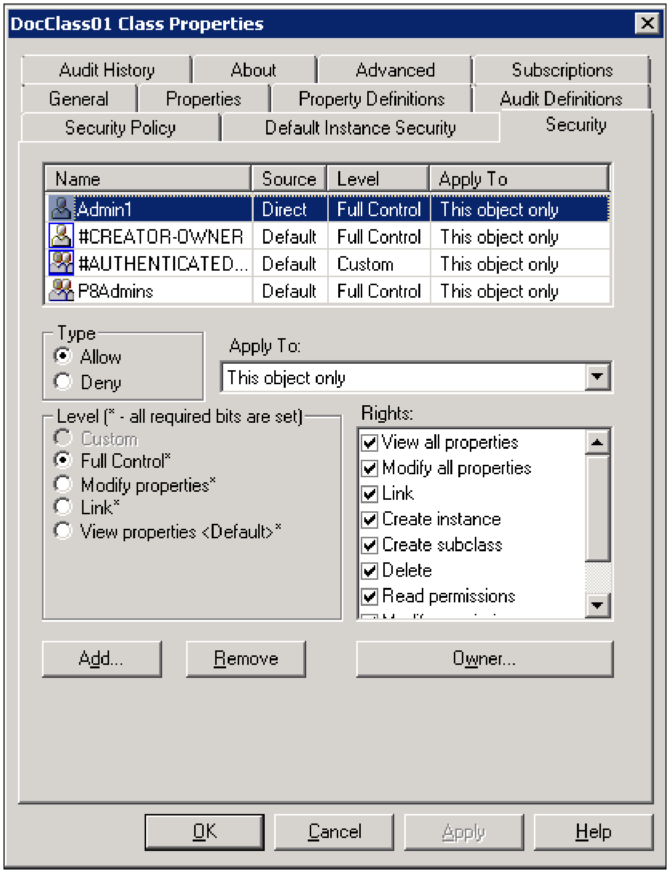
Any object within the Content Engine, from document class definitions to documents or custom objects to saved searches and folders, can have individual ACLs, which enable rich possibilities of predefined configurations to control inappropriate usage of these objects. Objects that shall not be used by unauthorized (relative to their access role) users, are not displayed or accessible without appropriate access rights.

The ACL of each securable object can be reviewed and defined by a security editor. The FileNet Enterprise Manager uses the security editor in the following three security tabs:

- Security: Displayed as part of the property sheet of all securable objects. (8.3.1, “Security of Content Engine objects” on page 215)
- Default Instance Security: Displayed as part of the property sheet of all securable classes. (8.3.2, “Default instance security” on page 221)
- Template Security: Displayed when the users accesses the General tab of a security policy and clicks **Modify**. (8.3.3, “Security precedence and inheritance” on page 223).

8.3.1 Security of Content Engine objects

To assign individual access rights or permissions of a current instance of a Content Engine object, the Security tab of the properties’ UI within the FEM can be used, as shown in Figure 8-7 on page 216.



Name	Source	Level	Apply To
Admin1	Direct	Full Control	This object only
#CREATOR-OWNER	Default	Full Control	This object only
#AUTHENTICATED...	Default	Custom	This object only
P8Admins	Default	Full Control	This object only

Type
● Allow
○ Deny

Apply To: This object only

Level (* - all required bits are set)
○ Custom
● Full Control*
○ Modify properties*
○ Link*
○ View properties <Default>*

Rights:
☑ View all properties
☑ Modify all properties
☑ Link
☑ Create instance
☑ Create subclass
☑ Delete
☑ Read permissions

Figure 8-7 Class property security tab

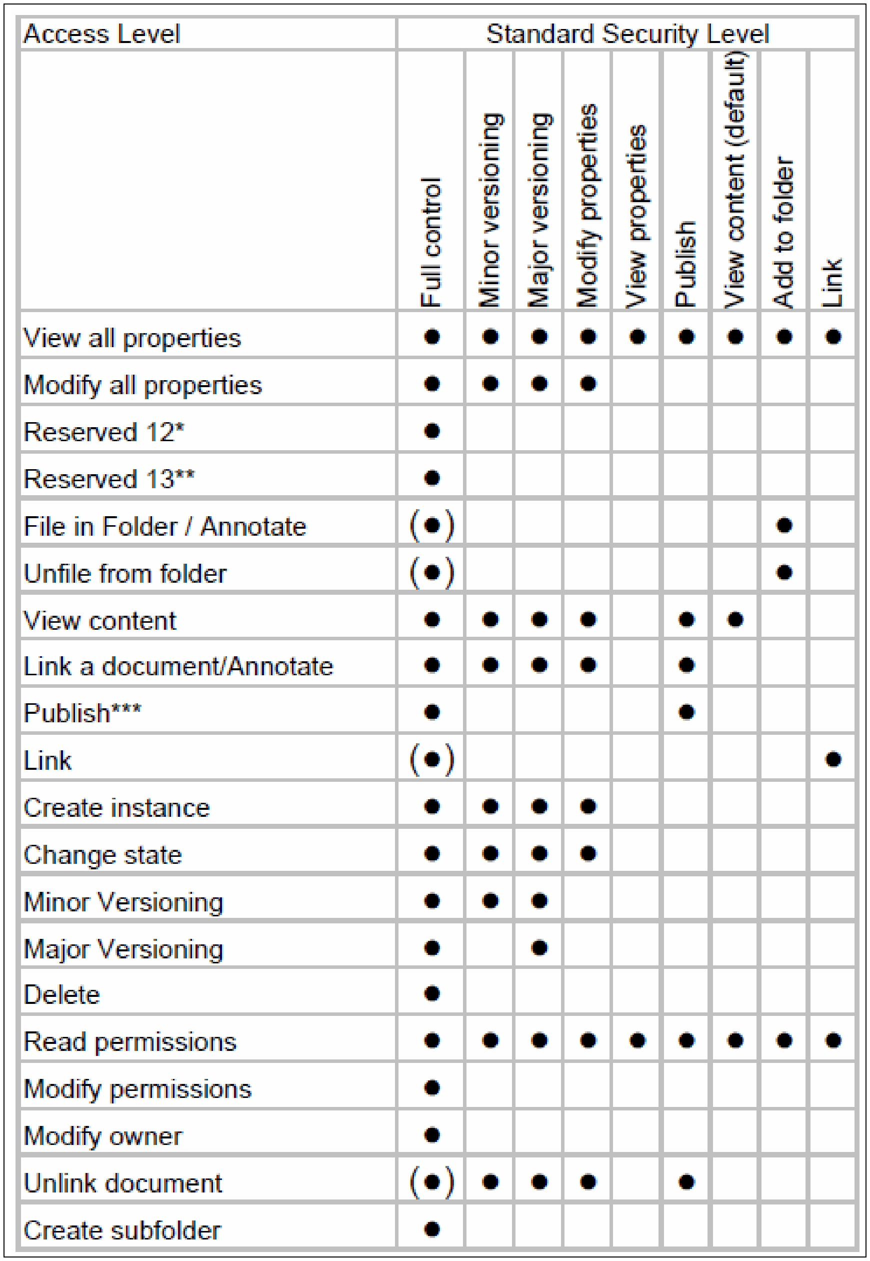
Standard security levels can be used to group permissions together in the administration interface, as shown in Figure 8-8 on page 218. As needed, customized access levels can be individually defined for any selected group or user account.

Table 8-2 provides the symbols that are used in Figure 8-8 on page 218.

Table 8-2 Symbols

Symbol	Description
•	Indicates that dedicated access rights are available for the given standard access level
(•)	Indicates that dedicated access rights are only available for a subset of standard access level

Symbol	Description
*	Indicates deprecation
**	Indicates that an archive is deprecated
**	Indicates that it is defined in Workplace to include Modify permissions



Access Level	Standard Security Level								
	Full control	Minor versioning	Major versioning	Modify properties	View properties	Publish	View content (default)	Add to folder	Link
View all properties	●	●	●	●	●	●	●	●	●
Modify all properties	●	●	●	●					
Reserved 12*	●								
Reserved 13**	●								
File in Folder / Annotate	(●)							●	
Unfile from folder	(●)							●	
View content	●	●	●	●		●	●		
Link a document/Annotate	●	●	●	●		●			
Publish***	●					●			
Link	(●)								●
Create instance	●	●	●	●					
Change state	●	●	●	●					
Minor Versioning	●	●	●						
Major Versioning	●		●						
Delete	●								
Read permissions	●	●	●	●	●	●	●	●	●
Modify permissions	●								
Modify owner	●								
Unlink document	(●)	●	●	●		●			
Create subfolder	●								

Figure 8-8 Overview of standard access rights level for Content Engine objects

Some of the rights in Figure 8-8 on page 218 are applicable to only some Content Engine objects. Table 8-3 is a list of which permissions are applicable to which objects.

Table 8-3 Object classes and permissions that affect access to them

Content Engine object class	Applicable permissions
Applicable to all	► View all properties ► Modify all properties ► Create instance ► Delete ► Read permissions ► Modify permissions ► Modify owner
Document	► Reserved 12 ► Reserved 13 ► View content ► Link a document ► Publish ► Create instance ► Change state ► Minor versioning ► Major versioning ► Unlink document ► create subfolder
Folder	► Reserved 12 ► Reserved 13 ► File in folder/annotate ► Unfile from folder ► Create subfolder ► Minor versioning ► Major versioning ► View content ► Change state ► Publish
Custom Object	► Link/annotate
Event action	► link a subscription
Other Classes incl. Task Task Relationship	► Link ► Create Subclass

Assigning permissions: It is possible to assign permissions to a Content Engine object that is not applicable to that class. This action is implemented to allow the inheritance of permissions from one object to another. See 8.4.4, “Dynamic security inheritance” on page 239.

A dedicated set of security configurations was established for Marking Sets, Object Stores, and P8 Domains.

Table 8-4 Content Engine object class and applicable permissions

Content Engine object class	Applicable permissions
Marking Sets Only Full Control or custom access levels available	▶ Add marking ▶ Remove marking ▶ Use marked object
Object stores Standard access levels Full Control (F) Use object store (U) View object store (V)	▶ Connect to store F,U,V ▶ Create new objects F,V ▶ Modify existing objects F,V ▶ Delete objects F,V ▶ Read permissions F ▶ Modify permissions F ▶ Modify certain system properties F ▶ Modify retention F
Enterprise Manager [P8 Domain] Standard access levels Full Control (F) Use stores and services (U)	▶ View all properties F,U ▶ Modify all properies F ▶ Create child objects F ▶ Delete child objects F ▶ Read permissions F ▶ Modify permissions F

Folders, Custom Objects, and Documents all have an owner property, which can be blank or point to a particular user or group. Whoever is assigned as the owner receives at least the following rights on the object:

- Read permissions
- Modify permissions
- Modify owner

By virtue of having the modify owner right, the owner also receives the read all properties right on the object; otherwise, the owner cannot see the owner property and therefore cannot modify it. If a user or group is given the modify any owner right on an object store, they are also given read all properties and modify owner rights on every object in the object store.

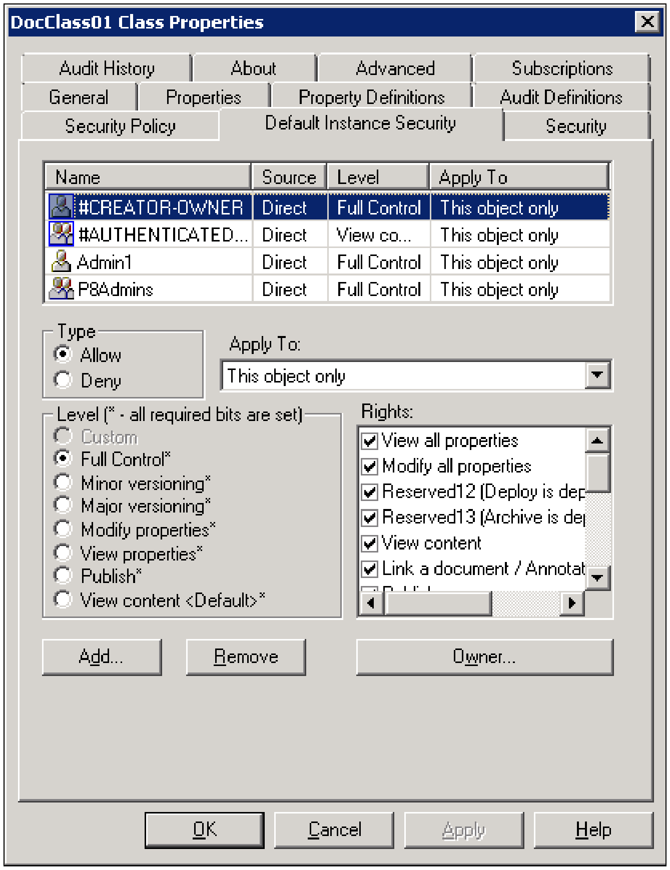
Markings and their constraints, which are discussed in chapter 8.4.1, “Marking sets” on page 229, are evaluated after the owner and object store permissions are assigned. As such, markings can be used to deny permissions that are granted to the owner of an object.

Assigning groups to the owner property: A group can be assigned to the owner property, which is useful where ownership of a document lies with a team rather than a user. Setting the owner to a group in this case gives anyone in that team the owner's rights. Over time, the group membership can be modified and no change to the owner field or access control list is required.

Evaluating the owner property: The owner property is evaluated after direct, template, and inherited permission sources. Any denial of those levels for the rights conferred by being an owner are overridden, and the owner still gets those permissions on the object.

8.3.2 Default instance security

Some securable object definitions can specify a *Default Instance Security* to apply to a new instance of this object when it is not provided by the application that is built on top of the Content Engine. Figure 8-9 on page 222 shows an ACL that is specified on a Content Engine document class object.



Name	Source	Level	Apply To
#CREATOR-OWNER	Direct	Full Control	This object only
#AUTHENTICATED...	Direct	View co...	This object only
Admin1	Direct	Full Control	This object only
P8Admins	Direct	Full Control	This object only

Type
● Allow
○ Deny

Apply To: This object only

Level (* - all required bits are set)
○ Custom
● Full Control*
○ Minor versioning*
○ Major versioning*
○ Modify properties*
○ View properties*
○ Publish*
○ View content <Default>*

Rights:
☑ View all properties
☑ Modify all properties
☑ Reserved12 (Deploy is dep
☑ Reserved13 (Archive is de
☑ View content
☑ Link a document / Annotat

Figure 8-9 Default Instance Security on a document class object

The following Content Engine objects allow to define Default Instance Security:

- Document classes
- Other Classes

If a subclass of this class is created, these ACEs are copied to the new class. However, updating a higher-level class does not automatically update subclasses.

Within these ACLs, two special entries can be used:

- The #CREATOR-OWNER allows rights to be assigned to whomever added the document (or whomever is specified in the Owner field at creation time.
- The #AUTHENTICATED-USERS entry can be used to refer to any user who is logged into the system.

#AUTHENTICATED USERS: #AUTHENTICATED USERS as in different group must *not* be used for authorization purposes (beside exceptions). It is always advisable to create a role and user concept that defines dedicated access levels for the usage of the Content Engine. Explicit authorization must always be preferred to implicit allowed access.

Belated changes to existing ACLs for numerous mounds of objects can generate significant system load. Besides this technical limitation, the risk of unauthorized access by accident cannot be obviated.

The Owner of a document instance can also be specified at the class level. By default, this is left as #CREATOR-OWNER, meaning that the owner of the document is specified as the user who created it. This rule can be overridden, however, to point to another user or group on the system. The overridden owner of the document can be useful when the owner must always be assigned to a particular user or group of users. If the #CREATOR-OWNER is specified to have particular permissions on a document instance, rather than on a class' default instance security, then these permissions have no effect. When the document is created, #CREATOR-OWNER is determined and that user is assigned rights on the new document instance. So, if user1 logged in and added a document, and the default Owner field is left as #CREATOR-OWNER, any rights assigned on the default instance security tab to #CREATOR-OWNER are assigned to the user1 on the new document instance. In other words, the #CREATOR-OWNER permission entry does not exist on the instance. It instead shows user1 as assigned to those permissions.

8.3.3 Security precedence and inheritance

The elementary nature of the FileNet P8 modules is that every access to a given object (folders, documents, inboxes) must be authorized, as discussed in previous chapters. The authorization restrictions range from basic authorization for all authenticated users to granular restrictions on a user basis.

It is quite unlikely for any user in an organization to have access to perform all operations on a document or other objects of the ECM environment. There are instances where all users might have read permissions, such as for policy documents, but generally, all content is locked down to some extent. This lock down can be done individually for each and every object, but this is cumbersome and hard to manage.

There are often cases where it is useful for security to flow through various objects to a document. The classic example is in foldering, where it might be required that the security on the document reflects that of the containing folder.

Another example might be a workgroup, such as an IBM Redbooks publication team, that specifies that a certain group of people have access to a set of documents. Perhaps the most classic example is security classification. These classification groups are created at an enterprise level and must be enforced to override or mask security permissions that are already on individual documents.

Before describing how authorization is calculated, the precedence of how these various security sources affect the target object must be discussed. The following list shows the highest priority security source first, followed by lower priority sources:

- Default or explicit (direct) Deny
- Default or explicit (direct) Allow
- Template Deny
- Template Allow
- Inherited Deny
- Inherited Allow

Permissions: A higher source of permission always overrides that of a lower source.

A source of default indicates that the security permission was assigned to the object through the default instance security mechanism that was previously mentioned. A direct permission is the same except it is assigned to the document instance directly rather than copied from the default instance security permission list. Both ways of assigning security result in explicit access rights on the given object.

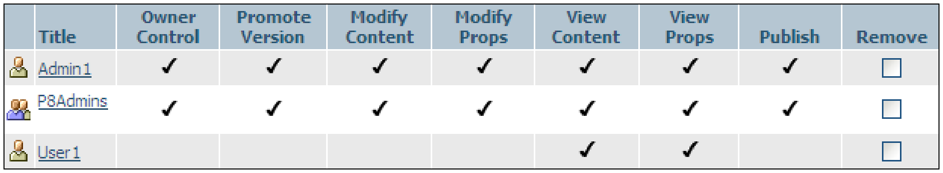
Security Templates are assigned to an object based on the settings within its assigned Security Policy. These are permissions that are copied into a document's access control list when the document's state matches a corresponding state in the policy that has a security template configured. When a match exists, the permissions are copied to the document.

Security template assignment: A security template assignment only happens when a life cycle event occurs or an application explicitly assigns an application security template and not when the security is evaluated.

This behavior is similar to that of a document lifecycle policy. The difference is that a document lifecycle state change occurs when a custom application calls one of the methods to modify a life cycle on a document, not in response to a versioning action, which is the case for security policies. A particular document lifecycle policy state might or might not be configured to apply security permissions. For more information about security and document lifecycle policies, see 8.4.3, “Document lifecycle policies” on page 237.

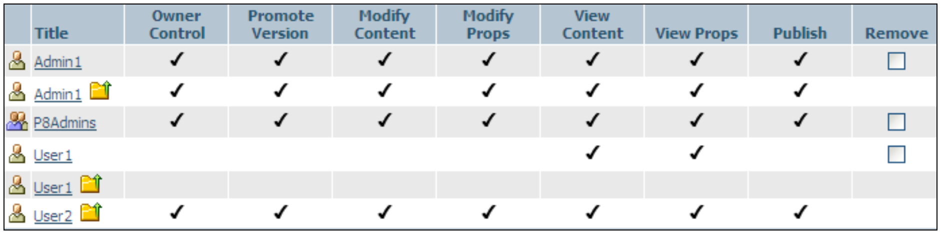
Inheritance specifies that the security is carried forward from another object, which is different from the previous approaches because the ACEs are not copied into the document but are dynamically evaluated on-the-fly. This method is particularly useful from a management perspective because many objects can inherit security from the same source object. This method is used by the Security Folder property on a document and folder to indicate the object from which security settings are inherited. It is also used in property-based dynamic security inheritance that is described in 8.4.4, “Dynamic security inheritance” on page 239.

Figure 8-10 shows the access rights for a given object assigned by the Default Instance Security. Figure 8-11 shows the same object after assigning additional rights by inheritance from a Security Folder. The inherited rights are marked by a small folder symbol and do not overrule the direct access rights.



Title	Owner Control	Promote Version	Modify Content	Modify Props	View Content	View Props	Publish	Remove
Admin1	✓	✓	✓	✓	✓	✓	✓	□
P8Admins	✓	✓	✓	✓	✓	✓	✓	□
User1					✓	✓		□

Figure 8-10 Standard explicit direct rights (sample)



Title	Owner Control	Promote Version	Modify Content	Modify Props	View Content	View Props	Publish	Remove
Admin1	✓	✓	✓	✓	✓	✓	✓	□
Admin1 📁	✓	✓	✓	✓	✓	✓	✓	
P8Admins	✓	✓	✓	✓	✓	✓	✓	□
User1					✓	✓		□
User1 📁								
User2 📁	✓	✓	✓	✓	✓	✓	✓	

Figure 8-11 Access rights with inherited rights from Security Folder (sample)

In Figure 8-11, User1 has explicit and direct View Content and View Properties rights and can view the content even having inherited Deny access rights. User2, in contrast, inherits the Full Control access rights from the Security Folder. The example demonstrates the ability to enhance rights and the inability to restrict access rights by using inheritance by Folder Security.

The missing check box in every row of inherited rights indicates that they cannot be removed. Inherited rights cannot be changed either (To open the security page, click the user name, but no changes to the ACE are possible).

8.3.4 Calculating authorization

Extra security ACLs for a document, folder, or custom object can be inherited. Security is calculated in the following order with each rule taking precedence over those that follow it in the list:

1. Explicit permissions that are assigned to the object instances are evaluated. A permission is marked as default if it is applied from the document class' default instance security settings.

   Explicit individual *Deny* authorization overrules implicit (for example, by group membership) *Allow access* rights.

2. Template permissions are evaluated. A template permission is assigned by a document lifecycle state change or security template. See "Security policies" on page 234 and "Document lifecycle policies" on page 237.

3. Inherited permissions are evaluated. These permissions can be from the Security Folder field or any Object Value Property that is defined with a Security Proxy Type of Inheritance. See "Dynamic security inheritance" on page 239:

   a. If a Security Folder is specified, it is checked for any ACEs that are specified to apply to the selected choice of entities: This object and immediate children or This object and all children. As of P8 5.0, the new additional inheritance entities' Immediate children, but not this object and All children, but not this object are available. An object can be filed in multiple locations but can have only a maximum of one Security Folder. This value is not set by the Workplace XT applications but by the means of the FEM. If the security folder is null (that is, not specified), no folder security is evaluated at any level.

   b. If this folder has its Inherit from security parent folder setting enabled, this folder's parents, starting with the immediate parent, are checked for all ACEs that apply to this object and all children.

   c. If all parent folders have Inherit permissions from parent enabled, eventually the root folder's security is checked, which is the top level of potential security inheritance for folders. By default, the root folder gives write permission to all users for all properties of any contained object in the object store. It is important to change this security on a production system.

4. The owner of the object is checked. If this is the user trying to perform the current action, this user is granted read all properties, read permissions, modify permissions, and modify owner rights on the object.

5. The containing object store's ACL is checked. Some object store permissions affect documents, folders, and custom objects. These permissions are *write any owner* and *privileged write*. The first of these permissions allows the

specified grantee to change the object owner. The second allows a grantee to modify certain properties on any object in the object store. These properties are creator, date created, last modifier, date last modified, and date checked in.

6. If the document has a property whose value is from a marking set, the specified marking is evaluated. If the current user does not have the use right on the marking, then the security permissions specified in the constraint are removed (masked) from the computed permissions list. Thus if a user has modify content rights but is not granted a marking's use right, the constraint can be configured to remove this right.

Note: On the General Tab of the properties of documents, as of IBM FileNet P8 4.0.1, the deprecated inheritance function Security Parent is offered. When selecting the **Inherit Security from folder** option in the pull-down box, all folders that the document is filed in are displayed to configure security inheritance. The purpose of this behavior is to support custom applications without change. The actual Security Folder property does not require containment of the document by the folder.

Refer to 8.9, “A practical example” on page 269, for a practical example that uses multiple sources of security, including inherited security, to illustrate how setting security at one level can override other security without causing any problems or gaps in the platform security landscape.

8.3.5 Authorization calculation example

Now an example for authorization will be discussed, assuming that there is a need to control the view content on a document instance. Table 8-5 shows the security settings that affect this particular document instance and its view content permission. Owner or object store granted permissions are not discussed to keep things simple. It is assumed, for the example, that none of the users are the document owner or have special rights from the object store permission list. The underlined permissions are those that have the highest precedence in our example.

Table 8-5 View content permissions and their sources for a sample document

Permission source	User A	User B	User C	User D
Direct Deny		Implicit deny	Y (Direct)	Implicit deny
Direct Allow	Y (Default)			

Permission source	**User A**	**User B**	**User C**	**User D**
Template Deny	Y (Document life cycle)			
Template Allow			Y (Security Policy)	Y (Security Policy)
Inherited Deny				Y (Security Folder)
Inherited Allow		Y (Security Folder)		
Marking use	Y	Y	Y	N
Marking applied constraint	N/A	N/A	N/A	Deny
Effective permission	*Allow*	*Allow*	*Deny*	*Deny*

Security for User A

This document's default instance security grants User A a view content permission on all new instances of the example document's class. User A is also explicitly denied this permission by a property that is assigned from a document lifecycle policy. An explicit direct permission has precedence over a template permission, which means that the default allow permission is still valid. User A is also granted the use right on a marking that is associated with the document, so the view content permission is not denied to User A by the marking constraint mask, which results in User A having an allow permission for view content on our sample document.

Security for User B

Neither the default instance security nor the document itself assigns User B a permission for the sample document, which results in an implicit deny because by default, the Content Engine denies permissions unless they are explicitly granted (as long as no general authorization by #AUTHENTICADTED USERS assigns access rights to all users). This document instance does have a Security Folder specified, explicitly granting User B the view content permission. User B also has use rights on the marking constraint mask, which results in User B having an allow permission for view content on the document.

Security for User C

User C has an explicit direct deny for the view content permission on the sample document. Even though an allow view content right exists from an applied security template, this has a lower precedence than the explicit deny. User C does have the use right on the marking, but this does not grant any extra rights to the document (a lack of the use right means the constraint mask denies the right). The effective permission is therefore to deny view content because of the explicit deny User C from the direct permission on the document.

Security for User D

No direct or default allow for view content is specified for User D; therefore, User D is implicitly denied the view content right for the same reasons that User B is. User D is, however, granted the view content right from an applied security template, so this takes precedence. The Security Folder has a deny view content permission entry, but this is of lower precedence than the allow view content from the security template. At this point, User D has an allow view content permission. User D does not, however, have the use right on the marking that is assigned to the document. As a result, the constraint mask is applied to User D's rights, denying User D view content rights on the document. Markings mask or remove individual permissions from a calculated permission list, so this is the setting that is applied. User D is denied view content rights on the sample document.

8.4 Setting security across the enterprise

When the default security for a particular class of object is known at design time, it can be assigned in the default instance security tab of the class. However, it is often the case that these security settings are common across document types and roles. This section describes techniques that allow security to be managed on a broader basis versus micro-managing every object's individual ACL. This process has advantages for both manageability and reduction of the database size due to access control entries. This section also describes how means of implicit security definition can support the configuration of a small instead of a huge number of objects to administer object security.

8.4.1 Marking sets

Marking sets provide a means to manage access to objects on a global basis through the values of specially defined *marking-controlled properties*, which are properties whose permitted values are drawn from the properties that are defined in the marking set and which influence the access check according to the definition of the Marking. Marking sets provide a generalized form of mandatory access control, also called *labelling*.

It is a fundamental nature of a marking set that access rights can only be denied and not allowed. Marking sets operate by having a list of values called markings, where each has its own list of grantees who are given the *use right* by the marking. A property can then be created on an object whose values are taken from a marking set, which works much the same way as choice lists except that choosing a particular marking value affects the security of the document.

The following limitations are to be considered when handling marking sets:

- Marking Sets are defined globally for a given P8 Domain
- Marking Sets are related to properties. As properties are defined within a dedicated object store, they are only active within the object store(s) and the related properties
- Marking Sets can only be defined when creating a property. They might be changed after creation, but not added to an existing property
- Marking Sets are not collocating with Choice Lists
- The number of Marking Sets within a single P8 Domain is limited to 100

Although the document instance security settings are still present, the marking constraints are applied or masked over these, which makes markings useful when stringent security that cannot be overridden must be applied across a number of document classes. A classic example of this is military or intelligence applications where there is a pre-existing security framework. A marking for a document can be created such that setting the value to Top Secret denies all users in lower security groups access to the object.

To use objects marked by marking sets, two separate conditions must be fulfilled:

- At least the use of the marking set related to a given property must be allowed.
- The configuration of the marking set itself cannot prevent the dedicated action on the document.

Markings: A marking differs significantly from other security methods used in IBM FileNet Content Manager. Normally, permissions that are given or denied from specified users and groups are explicitly listed. With a marking, however, only users and groups that have at least the *Use Right On* are required to be specified. If a user does not have this right, their computed permissions are masked (denied) by the constraint permissions that are enabled on the marking.

The use of marking sets can be restricted to:

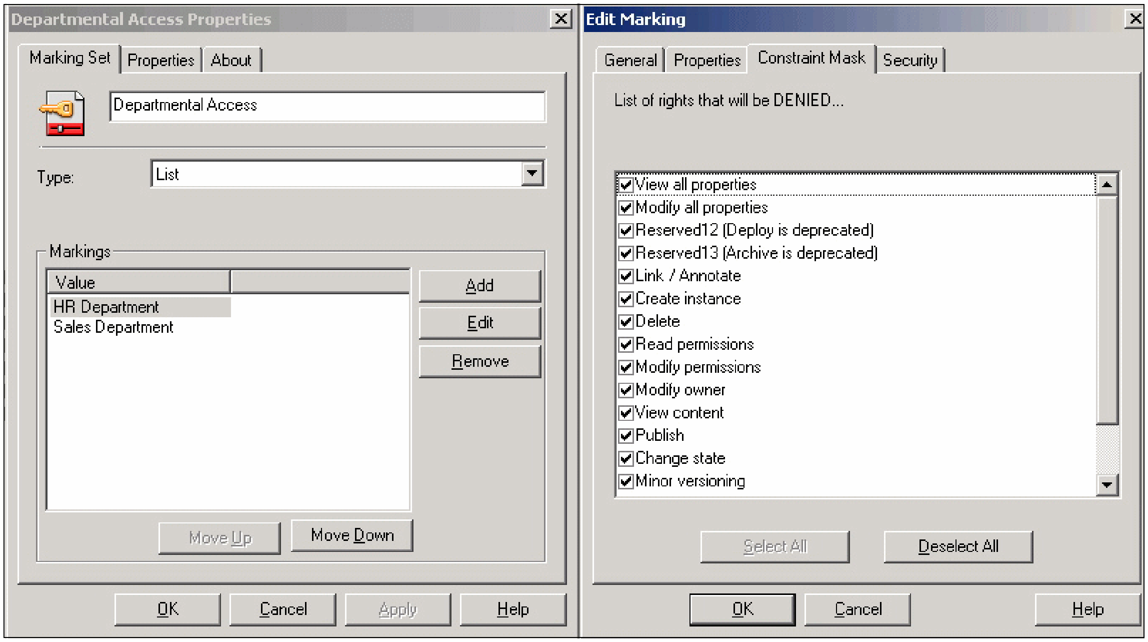
- Add Marking: Related values can be changed by the given user.
- Remove Marking: Related values can be deleted by the given user.
- Use Marked Objects: The related value assigns access rights to the given user s as defined in the related marking set.

If a given user does not have at least Use Marked Objects rights or has explicit Deny access rights on the marking set, the user is not allowed to use that object even if the explicit direct Allow access rights on the document.

There are two types of marking sets:

- Non-hierarchical: Apply only the constraints that are present on the selected marking value.
- Hierarchical: Apply all lower marking use rights prior to its own, which means that, for the previous example, the Top Secret marking only holds additional entries for users with Top Secret level access rather than re-list all the entries for Secret, Confidential, Restricted, and Public. This makes administration much easier and prevents oversights in security from occurring.

For clarification shall the following example serve as a model. It shall be necessary to ensure that all non-HR users cannot read the properties or content of all documents that have a department marking property of HR. A marking set is created called Department Set with several markings, one of which is called HR Department. It is specified, that the use right is given to the HR Department group as held in the directory server. A constraint mask for all of the permissions that shall be denied is added. In Figure 8-12 on page 232, notice that all permissions to groups other than HR are denied.



Departmental Access

Type: List

Markings

Value	
HR Department	
Sales Department	

List of rights that will be DENIED...

☑ View all properties
☑ Modify all properties
☑ Reserved12 (Deploy is deprecated)
☑ Reserved13 (Archive is deprecated)
☑ Link / Annotate
☑ Create instance
☑ Delete
☑ Read permissions
☑ Modify permissions
☑ Modify owner
☑ View content
☑ Publish
☑ Change state
☑ Minor versioning

Figure 8-12 The marking set properties and marking properties dialog boxes

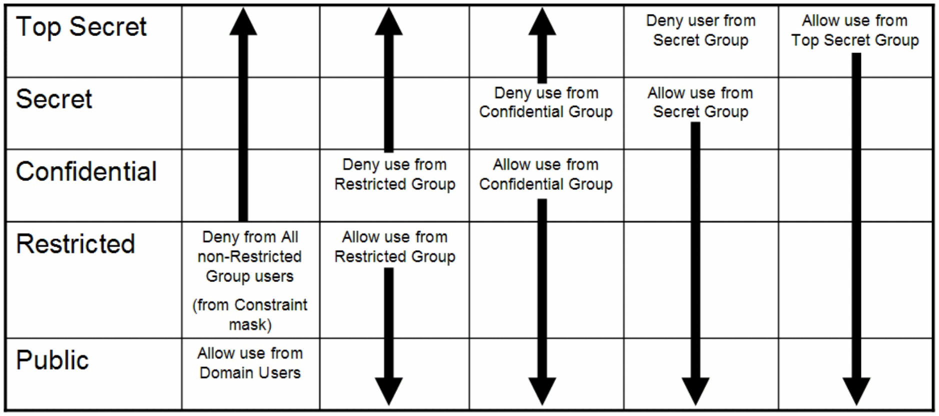
If a non-HR user tries to access the document, the user is prevented from reading the document's properties and content by virtue of the user not being in the HR Department group.

Implementing the equivalent security restrictions using permission lists requires specific enumeration of all groups that must not view content and properties, making security management across the organization far more difficult than using markings.

Markings do not exist at the object store level. They are configured on the IBM FileNet P8 Domain and as such can be re-used by any object store. This behavior must be considered when updating a marking that this has instant effect across the enterprise and not compulsive on a single object store. Multiple markings can be applied to the same document, which causes all constraint masks that are specified in the active markings to be enacted cumulatively.

Figure 8-13 on page 233 shows a sample security classification. A hierarchical marking set is created so that all access that is denied at the Restricted level is carried on to the Confidential, Secret, and Top Secret levels. Remember that a marking set propagates only the Add Marking, Remove Marking, and Use Marked Objects permissions. Permissions are only masked (and thus denied) by a marking if its constraints mask has specific permissions that are listed to deny.

An hierarchical set that propagates Deny rights and implicit Allows (by not denying and assuming that the appropriate right was granted by direct ACL) in Figure 8-13.



Top Secret	↑	↑	↑	Deny user from Secret Group	Allow use from Top Secret Group
Secret	↑	↑	Deny use from Confidential Group	Allow use from Secret Group	↓
Confidential	↑	Deny use from Restricted Group	Allow use from Confidential Group	↓	↓
Restricted	Deny from All non-Restricted Group users (from Constraint mask)	Allow use from Restricted Group	↓	↓	↓
Public	Allow use from Domain Users	↓	↓	↓	↓

Figure 8-13 Use permission propagated in security classification marking set example

As shown in Figure 8-13, hierarchical propagation has the effect of *deny rights being propagated upwards* and implicit *allow rights being propagated downwards*. In other words, if a user has use rights at the Secret level, they also receive use rights for all documents that are marked Confidential, Restricted, and Public.

On all markings other than Public, all permissions are specified in the constraint mask, which denies access to any object from anyone in the system who does not have the Use permission on the marking. For Public, an Allow use for all domain users was added. In our setup with Active Directory, all users are members of the domain users group. Consequently, no constraint mask for Public are added because it is not evaluated.

Markings: Markings allow content to be hidden from system administrators.A marking can be specified to deny the Modify Owner right to all users who must not change a document, folder, or their security. The administrator can be included in this group. The Use Marked Objects right can then be assigned to all other users who need to change the document, folder, or their security. Adding a required property that uses this marking to the top level Document class (or Folder or Custom object) with the default value of the specific marking created enforces the marking for all new documents. In this way, administrator access is denied to any content within the system, while still allowing them to configure the system.

Expansion products use the core functionality of the IBM FileNet P8 Platform and apply these capabilities to new problem domains. Marking sets are used extensively in IBM Enterprise Records to lock down content that is declared as a record. This marking prevents any user who is not a member of the Enterprise Records group from deleting or modifying the content and properties of any document version that is declared as a record. Setting marking set values can be a manual step, such as when an investigator accesses the document or an automated step during the records disposition process, checking, moving (to long term storage), or deleting a record during its life cycle.

This approach provides several advantages. The security mechanism in use is familiar to existing IBM FileNet Content Manager or IBM FileNet Business Process Manager (BPM) administrators, which makes it easier for administrators to start using an expansion product. Many of the groups and access control specifications might already be in place or can be re-used for an IBM Enterprise Records implementation. A customer also gets the added assurance of knowing this security feature was extensively tested in IBM FileNet Content Manager (or BPM) implementations.

Markings: Documents can have multiple markings. The effective constraint mask is calculated with all cumulative denials being applied.

It is also possible that a marking property can have more than one value. A document can, for example, be applicable to multiple departments, but might not be allowed to be accessed from anywhere else. Although it is allowed to assign hierarchical *Marking Sets* to multi-value properties, this is not recommended. Because an element in a hierarchical set inherits settings from lower precedence markings, it does not make sense to assign multiple hierarchical markings to the same property.

8.4.2 Security policies

Security policies provide a mechanism for automating changes to an object's ACL as its state changes, either through versioning operations or application-controlled state changes.

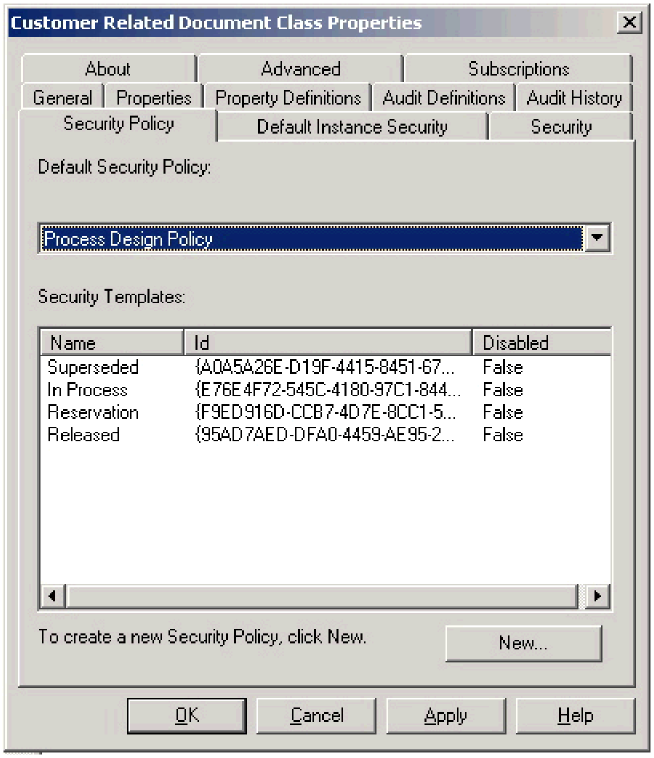
Security policies consist of security templates. These templates contain ACEs, just like an object instance's security settings. There are two types of security policy templates: *version templates* and *application templates*. A version template is assigned to a document when its versioning state changes (for example Released, In Process, Reservation, or Superseded). An application template, by contrast, is assigned by a custom application, explicitly setting the template through an API call.

This behavior makes security policies more decoupled from the document's properties. Permissions to apply do not need to be explicitly stated but rather change with the document's state. After this change is made, the permissions on the applicable security template are copied onto the document version. They are not dynamically resolved like they are for marking sets or dynamic security inheritance properties.

Whether the security policy adds the applicable template's entries to the document's security permissions or whether it completely replaces them can be specified. This option is particularly useful because each security template only needs to contain the changes to the document instance security that must be applied.

Security policies can be best thought of as objects that modify a document's security during its life cycle rather than as explicitly overriding security based on some manual action.

A default security policy can be assigned in the class definition settings dialog, as shown in Figure 8-14 on page 236. They can also be assigned to a specific version of a document later in its life cycle. However, changes to the current version's security policy is only processed the next time it is versioned because that is when the security templates are checked and applied. They cannot be applied to the current version and be expected to change the document's security immediately because that does not happen until some versioning state event occurs, such as *Checkout*.



Default Security Policy: Process Design Policy

Security Templates:

Name	Id	Disabled
Superseded	{A0A5A26E-D19F-4415-8451-67...	False
In Process	{E76E4F72-545C-4180-97C1-844...	False
Reservation	{F9ED916D-CCB7-4D7E-8CC1-5...	False
Released	{95AD7AED-DFA0-4459-AE95-2...	False

To create a new Security Policy, click New.

Figure 8-14 Assigning a default security policy to a new instance of a class

In addition to the four default document states, custom or application states can also be created, which are useful in situations during a document's life cycle, during a managing process, or when it is known at design time who has access to a document.

Considering the example of a software company that produces and publishes documents, which are called technical papers. A technical paper can go through an authoring state at a minor version (such as 1.1, 1.2, and 2.4) and at an editorial state. After it is checked by an Editor, it can be promoted to a major version that is ready for publication. The security for these steps can be modified using versioning security templates. Considering that a member of the Legal team also checks the technical paper before it is published. The Legal team has access to the released version, thanks to a versioning security template. After they check the technical paper, they move its state to the Ready for Publication application state. This state prevents any further modification to the document and requires a custom application state to be specially created and applied to the technical paper class.

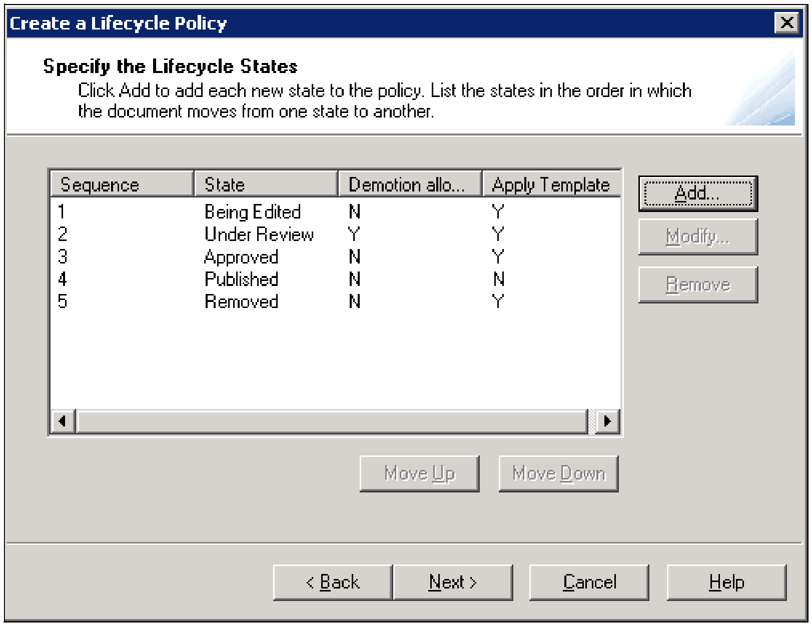
Using this method, technical papers are moved through their application-specific life cycle, modifying security along the way as most appropriate. This progressive movement is accomplished by configuring document states and security templates within the Content Engine and the process required no business process.

Of course there are situations where this approach falls short. An example might be, when security is not assigned to a group of people with a role, but rather it must be dynamically decided. This type of requirement is best accomplished within a business process, which is described in 8.5, “Gradual security requirement changes” on page 250.

There is also the issue where only one security policy can be assigned to a document at a given time. In certain situations, someone might want to add extra security to a document. An example might be adding a restriction on who can see a document by country because of specific local legislation. In this case, it is best to use a Marking Set or a Dynamic Security Inheritance Object, depending on the situation.

8.4.3 Document lifecycle policies

A mechanism similar to that provided by security policies is available for automating ACL updates as a document moves through the states defined by a document lifecycle policy. It is possible to create document lifecycle policies to have more granular control over how a document is accessed and worked on during its existence. If a simple document creation is considered, approval, and publishing example, a document life cycle can be used to set the visibility (view content permission) of a document at specific phases in its life cycle, as shown in Figure 8-15 on page 238.



Click Add to add each new state to the policy. List the states in the order in which the document moves from one state to another.

Sequence	State	Demotion allo...	Apply Template
1	Being Edited	N	Y
2	Under Review	Y	Y
3	Approved	N	Y
4	Published	N	N
5	Removed	N	Y

Figure 8-15 A simple document editing, approval, and publishing life cycle

Security can optionally be assigned when a document transitions to a particular state. When going through approval and (re-)editing, the author's edit content rights might be temporarily removed and then added back again if a change is required. Not all states need to change security.

After a document is approved, all versioning and modification permissions are removed from both the content authors and content approver in the previous example. No further security modifications are needed until, for example, the document needs to be checked out for an update, so the Published template does not apply any security of its own.

It might be necessary to perform some system action based on changes in states, which is done by assigning a custom Java class as a life cycle action that is invoked whenever the state changes. For the example, it might be necessary to file the document in different locations after it is published, and then automatically move the life cycle on to the published state.

Any permissions that are assigned by a life cycle policy have a source of template and are therefore evaluated at the same level as security policies. A life cycle policy can be assigned to one or more document classes, but each class can only have one life cycle policy.

Document lifecycle policies can be useful if the processing requirement is simple and a full business process management solution is overkill.

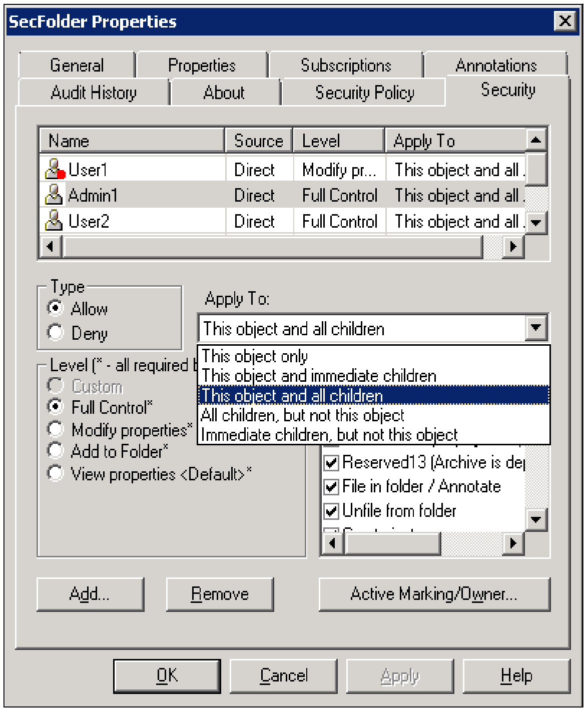
8.4.4 Dynamic security inheritance

As mentioned in chapter 8.3.3, “Security precedence and inheritance” on page 223, the concept of security inheritance is a strong means for security design in IBM FileNet P8 environments. Rather than having a property value determine additional security on a document, the property, itself, specifies another Content Engine object from which to inherit the security settings.

In the previous chapters, explicit security settings set by default instance security or by configuring directly on the given object were described. They are not changed when the relative security source configuration is changed. This chapter describes means of security inheritance that dynamically change the ACLs of the related object whenever changes are made on the source of the inherited security settings.

The object that inherits the security still has its own direct security settings and they are not modified using this method, but they are supplemented by any inherited permissions. If a Content Engine object has direct, default, or template permissions, however, they have a higher precedence than inherited permissions.

For each ACE that is used for inheritance, five separate depths exist that must be configured individually. For the security definition, the security tab of the properties of the object to inherit from is used within the FEM, as shown in Figure 8-16 on page 240.



Name	Source	Level	Apply To
User1	Direct	Modify pr...	This object and all .
Admin1	Direct	Full Control	This object and all .
User2	Direct	Full Control	This object and all .

Type
● Allow
○ Deny

Level (* - all required b
○ Custom
● Full Control*
○ Modify properties*
○ Add to Folder*
○ View properties <Default>*

Apply To: This object and all children

This object only
This object and immediate children
This object and all children
All children, but not this object
Immediate children, but not this object

☑ Reserved13 (Archive is de|
☑ File in folder / Annotate
☑ Unfile from folder

Figure 8-16 Possible configuration selection: security inheritance

In Figure 8-16:

- This object only: This ACE takes effect in access checking for the current object, but is not inherited by any children.
  This is the default setting for new ACEs.
- This object and immediate children: This ACE applies to the object and is inherited by the parent object's children, but not by the child object's children. After inheritance takes place, the child ACE has an inheritable depth of This object only.
  Use this option only if there is no structure beneath the given inheritance.
- This object and all children: This ACE applies to the object and is inherited by every generation of the parent object's child objects. After inheritance takes place, the child object's ACE has an inheritable depth of this object and all children.
  Chose this option whenever the usage of the parent object and all its descendants are used with a similar configuration.
- All children but not this object: This ACE is inherited by every generation of the parent object's child objects, but does not affect the parent object itself.
  Use this option if the inherited security of a given group (or user) is different of

the parent object. This results, for example, when objects used for security inheritance need separate administrative access rights for the security parent and the security children.

- Immediate children only, but not this object: This ACE is inherited only by the parent object's immediate children, but not by further generations, and does not affect the parent object itself.
  Use this option if the inherited security of a given group (or user) is different for use of the parent object itself and it is conceivable that no additional level of inheritance is needed.

The relationship between an inheriting object (the child) and the inherited-from object (the parent) is established through an object-valued property (OVP) of the child configured with a SecurityProxyType *inheritance*. The value set for such a property identifies the specific parent instance from which the child object inherits. A class can define multiple such properties, and in that case, the inherited ACEs for an instance of that class is the union of the inheritable ACEs drawn from all of the objects referenced by those properties.

Several system properties have the inheritance proxy characteristic and so provide out-of-the-box inheritance behavior. This object must have at least one inheritable ACE. For configuring security inheritance by using a custom object-valued property see:

http://publib.boulder.ibm.com/infocenter/p8docs/v5r0m0/index.jsp?topic=/com.ibm.p8.security.doc/p8psh008.htm

> **Note:** If the object class with OVP of a given object is changed to an object class without OVP, all inherited access rights are lost!

Table 8-6 on page 242 summarizes how objects receive initial security. Security inheritance takes place only if the source class or object has inheritable permissions.

Table 8-6 Relationships between security parents and their children

Object	Initial security comes from...	Inherits additional security from...	Its security can be inherited by...
Folder	The DefaultInstancePermissions of its class, or directly set when creating. Security policy, if configured.	Its parent folder (the folder immediately above), if the Inherit parent permissions check box is selected on the child folder. Custom object-valued properties with Security Proxy Type set.	Child folders, if Inherit parent permissions is enabled on those child folders, and if there are inheritable ACEs. Documents or custom objects that consider the folder its security folder, if the folder has inheritable ACEs. By other objects (document, custom object, folder) through a Security Proxy Type property and acting as security parent.
Document	The DefaultInstancePermissions of its class, or directly set when creating. Security policy, if configured.	Security folder, if configured using Security Parent or Security Folder properties. Custom object-valued properties with Security Proxy Type set to "Inheritance".	Any annotations assigned to the document version, if the document has inheritable ACEs. By other objects (document, custom object, folder) through Security Proxy and acting as security parent.
Custom object	The DefaultInstancePermissions of its class, or directly set when creating. Security policy, if configured.	Same as Document.	By other objects (document, custom object, folder) through Security Proxy and acting as security parent.
Annotation	The DefaultInstancePermissions of its class, or directly set when creating.	Document, Folder, Custom Object.	None.
Other classes	Its parent class.	Any additional parent classes up to the top of the class hierarchy.	Child classes, if there are inheritable ACEs.

Dedicated folders, where every child had to be filed in the folder, formerly used as parent object for folder security inheritance, were deprecated as of IBM FileNet P8 4.0.1. This special folder ability as *Security Parent* remains only for compatibility purpose and was replaced by the described folder inheritance algorithms.

Inherited access rights: Inherited access rights cannot be configured separately on a user or group basis if related to a given object. If the inheritance is based on a security folder, this characteristic can be deleted from an earlier version of a given object without impact on existing later versions. The remaining latest version still inherit its current inheritance configuration and changes on the parent object immediately change the inherited child security.

Deleting the characteristic “inherit from security folder” must be executed using the FileNet Enterprise Manager and is not available within the standard UIs as Workplace XT or Widgets.

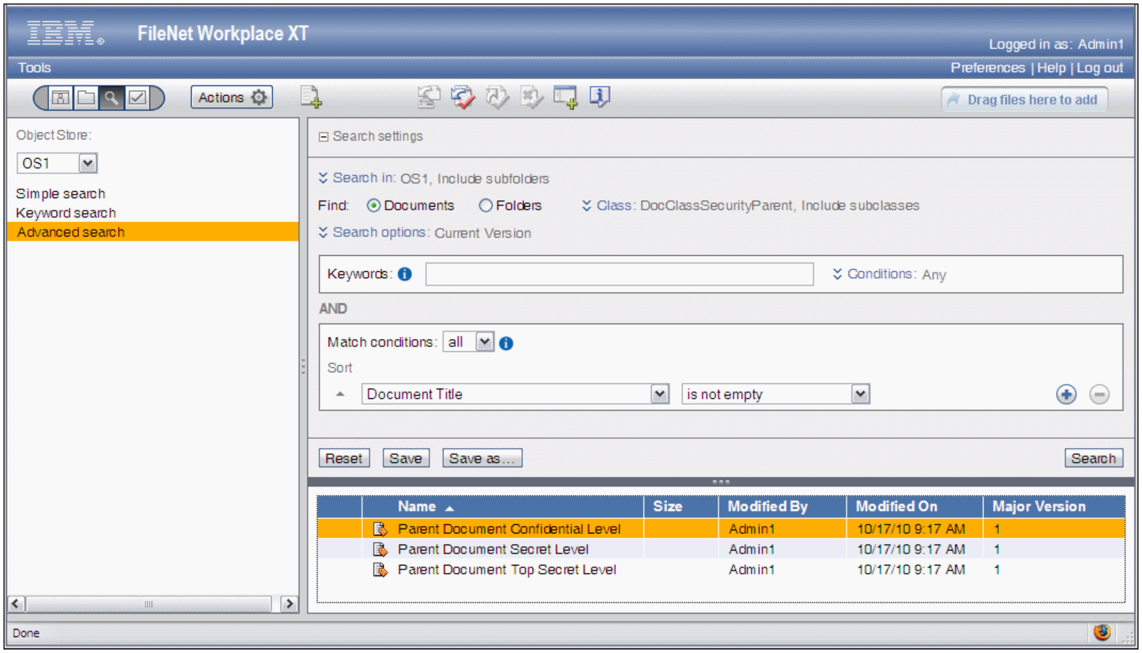
Together with the previously mentioned inheritance depths the means define disposition and possibilities of security inheritance. Detailed configuration descriptions are available at the IBM FileNet P8 Information Center in the Security section of the subsection Configure Security Inheritance. See also:

http://publib.boulder.ibm.com/infocenter/p8docs/v5r0m0/index.jsp?topic=/com.ibm.p8.security.doc/p8psh005.htm

Tip: Creating a dedicated class or classes for managing inheritance is recommended in complex environments to provide an administrative advantage because these parent objects can easily be found by searching on the dedicated class.

Further on it might be helpful to use document and folder objects as parent objects for inheritance because they can be named in an easy-to-read manner.

Figure 8-17 on page 244 and Figure 8-18 on page 245 show the sample queries based on the previously-mentioned recommendation in Workplace XT and FEM respectively. This is an easy way to find security parent objects when they are organized within a single class (or a small number of classes). At the end of this chapter, 8.9, “A practical example” on page 269, gives a more detailed example.



Object Store: OS1

Search settings

Search in: OS1, Include subfolders

Find: ● Documents ○ Folders

Class: DocClassSecurityParent, Include subclasses

Search options: Current Version

Keywords: Conditions: Any

AND

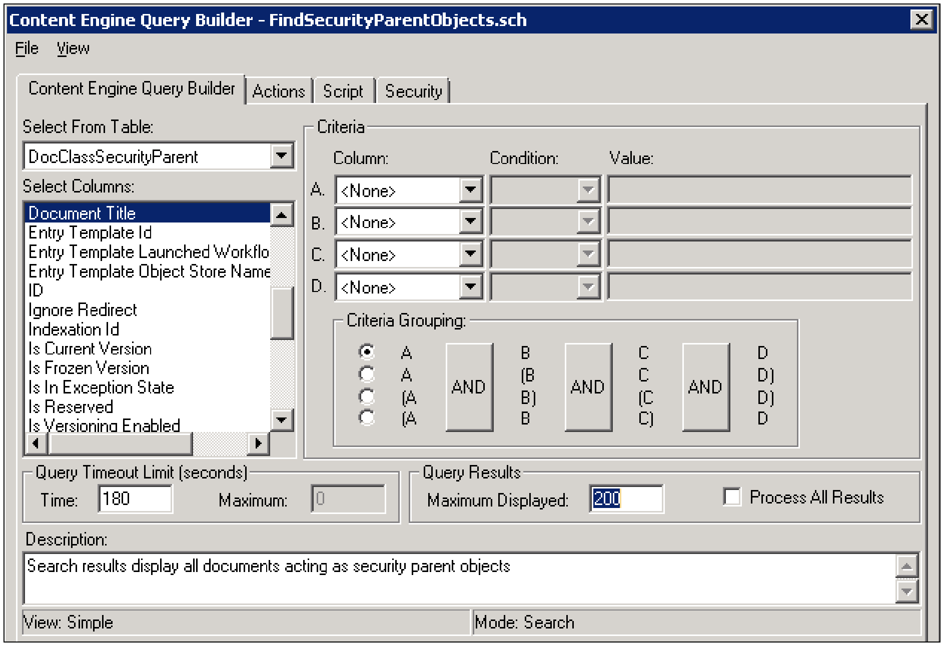
Match conditions: all

Sort

Document Title is not empty

Name	Size	Modified By	Modified On	Major Version
Parent Document Confidential Level		Admin1	10/17/10 9:17 AM	1
Parent Document Secret Level		Admin1	10/17/10 9:17 AM	1
Parent Document Top Secret Level		Admin1	10/17/10 9:17 AM	1

Figure 8-17 Search query and result for sample documents acting as security parents in Workplace XT



Content Engine Query Builder - FindSecurityParentObjects.sch

Select From Table: DocClassSecurityParent

Select Columns:

Document Title
Entry Template Id
Entry Template Launched Workflo
Entry Template Object Store Name
ID
Ignore Redirect
Indexation Id
Is Current Version
Is Frozen Version
Is In Exception State
Is Reserved
Is Versioning Enabled

Criteria

	Column:	Condition:	Value:
A.	<None>		
B.	<None>		
C.	<None>		
D.	<None>		

Query Timeout Limit (seconds)

Time: 180 Maximum: 0

Query Results

Maximum Displayed: 200 □ Process All Results

Description:

Search results display all documents acting as security parent objects

View: Simple Mode: Search

Figure 8-18 Search query for sample documents acting as security parents in FEM

8.4.5 Security-partitioned systems

It might be necessary to have systems that must be strictly bordered from each other. The reasons can be that the systems have multiple tenants with highly restricted contents or for compliant reasons. This section discusses various ways to partition systems under security aspects.

Recapitulating security instruments

The purpose of creating an *ethical wall* (previously known as Chinese wall) within a company's environment is to ensure that two internal groups using the same IBM FileNet P8 system do not see each others' content, which occurs in competitive situations where an organization might be subcontracted by two or more competing line-of-businesses and must show compliance by securing all sets of content separately. This scenario can include design information, legal disclaimers, problem reports, or contractual amounts. Another scenario is the need of data separation for multi-tenancy purpose.

In such cases, it is vital that information is restricted only to the delimited users of the related given isolated data. For administrative activities, there are additional needs for system access without necessarily notice of content like document display.

There are different possibilities to build up complex security frameworks. Table 8-7 illustrates several general approaches as an assistance for the development of a security model beyond direct security definition on individual objects. Finally, the table summarizes major aspects of the previously-mentioned security capabilities.

Table 8-7 Security configuration general approaches

Mean of security configuration	Capable for	Less capable or unsuitable for
Organizational techniques		
Dedicated P8 Domains	► Complete independent security models realizable, even for administrative purpose on P8 Domain level	► Complete independent security models, accordingly administrative overhead necessary
Dedicated object stores	► Separate data access for multi-tenancy, different lines of business, and so on ► Dedicated security models within a particular object store ► If necessary, dedicated data bases per object store ► Resolve concurrent security requirements ► Full capability of all object store related security means ► User access can be limited to single Object Stores	► A strongly increasing number of object stores result in a highly-increasing complexity of the given FileNet P8 Domain. Dependence on the individual complexity is the reasonable number of object stores to be reached at 10...20 object stores. However complex frameworks with clearly more than 100 object stores have been realized successful. ► No reduce of security complexity ► No magic bullet for any security configuration issue
Security instruments		

Mean of security configuration	Capable for	Less capable or unsuitable for
Marking sets	► Enhancement of existing direct or inherited object security ► Deny access right by marking sets overrule existing explicit or implicit allows ► If dedicated functionality shall be declined while other functionality shall stay usable (e.g. if for administrative purpose documents shall only not be displayed) ► Changes on the marking sets have immediate impact on the objects they are used by	► No additional allow rights on the object's security ► Data separation for multi-tenancy purpose only by marking sets not possible ► Mapping of complex allow and deny configurations ► Limited to ca. 100 marking sets per P8 Domain
Static configuration by default instance security	► Automatically provided predefined ACLs at creation time ► Offers a fine granularity of direct and explicit Allow and Deny access rights ► Can be manually changed after creation, when individually needed	► No automated changes when external changes (on the perspective of the given object) take place ► Changes to a large number of ACLs is difficult, if necessary
Quasi static configuration by security policies or lifecycle policies	► Enhancement of existing direct or inherited object security ► Versioning event used for security changes ► Predefined changes concerning the security alongside the lifecycle of an object possible	► No automated changes, when external changes (on the perspective of the given object) take place ► Active changes to a large number of ACLs is impossible without versioning and for all objects without relation to any security policy ► Existing ACLs—even when created by a security policy— are not affected by changing the security policy

Mean of security configuration	Capable for	Less capable or unsuitable for
Dynamic configuration by inheritance	► Changes on the parent object with immediate impact on the children objects ► Enhancement of existing direct or object security	► Direct Allow overrules inherited Deny ► Direct Deny overrules inherited Allow

Figure 8-19 shows a sample security architecture that encircles two data areas with a common administration.

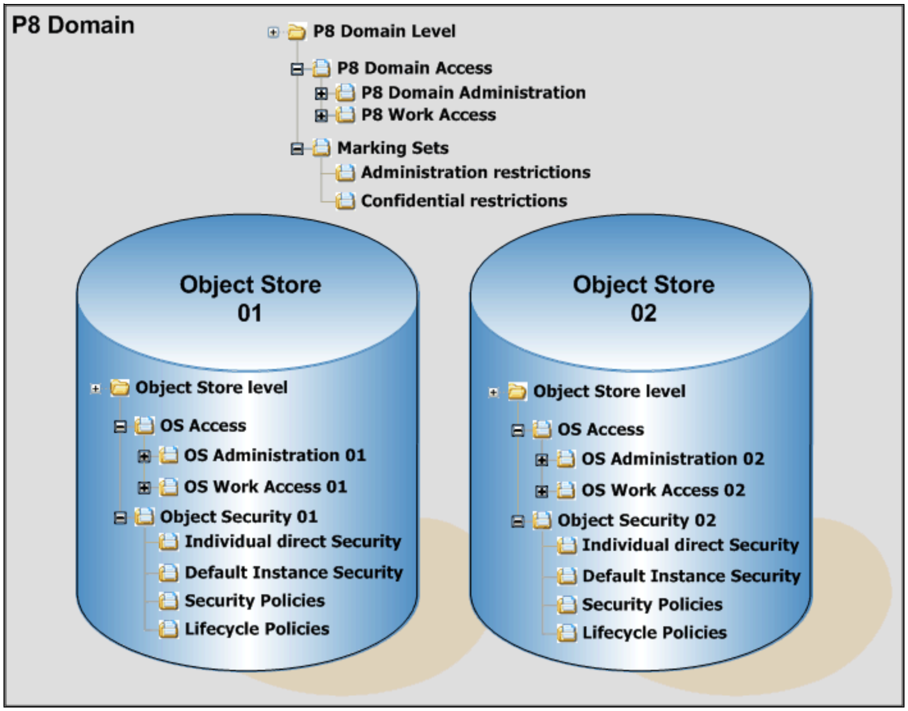


Figure 8-19 Sample Object Store separation

The overall concept of having a data (and security) separation, as shown in Figure 8-19, is to have:

- A common administration for the P8 Domain including the access right to define Marking Sets for administrative and highly restricted access rights in all Object Stores where appropriate.

- An individual *Object Store* (OS) security model for object access rights for various levels of OS specific groups and security means.

As a result, two or more widely-independent Object Stores can be operated. For separation purposes, non-administrative user are only enabled, by security restrictions, to use dedicated Object Stores. Administrative users can be restricted (for example, Deny access rights on View Content for all documents) highly granularly to fulfill all necessary administrative tasks. Enhanced administration can exceed splitting administrative tasks in P8 Domain wide and Object Store specific tasks. If the enhanced administration still is not applicable, only dedicated P8 Domains are appropriate. Based on the limitations of Marking Sets, they are no means for general security management but instead for dedicated tasks where particular restrictions are appropriate.

The #AUTHENTICATED USER ACE is not recommended for complex security configurations; instead, they must be deleted in an early stage of system configuration to prevent accidentally remaining security ACEs.

> **Security configuration:** The security configuration must always be kept as simple as possible and as granular as essential. Increasing granularity activates proliferates administrative overhead in planning and maintaining the security.

Examples of using complex security configuration

Many organizations purchase emerging technologies at the departmental level until a critical mass is reached and it makes sense to deploy the technology centrally. A classic example of this is database applications. Many organizations started deploying databases as stand-alone storage mechanisms for application information and configuration. Databases gradually became more ubiquitous and managing the information, security, backups, and resilience to failure became a major challenge.

Most organizations now have central teams to manage mission-critical software, such as a database farm. As Enterprise Content Management systems become more common, the same trend of adoption can be seen. It might be that forward thinking departments are setting up a team to look after a service even while it is deployed at their department's level. This action enables them, in the future, to allow other users in the wider organization to have access to the same information, creating a single source for information stores. A classic example of this is customer information that is managed in separate departments, such as email support correspondence, account opening information, and billing statements.

It might also be the case that an organization wants to create a shared service while also maintaining independent, discrete sets of information. A fraud investigation team, for example, might need access to the entire organization's set of customer information, while separately maintaining its own secured repository for investigation reports. Such scenarios require a solution at the object store level, as described in 8.4.5, “Security-partitioned systems” on page 245, whereby such investigative users might act in an administrative role in one Object Store and as a single user role in another.

By using the same IBM FileNet P8 Domain for multiple applications, it is possible to not require additional server and software rollouts. Creating an extra object store requires the creation of a new database table space and configuration of some data sources. Dedicated Databases might be used. This process offers the full benefits of a secured repository that uses corporate-wide security settings from marking sets while implementing new application specific settings. It might also be that the wider organization must not see certain types of information, represented by Content Engine properties, that is stored with the documents. Creating a separate object store can hide the fact from the wider organization.

A separate object store is also useful when calculating how much resources departments are using at both the database and file storage levels. Because the entire application has its own storage locations, calculating the space used and the amount to charge that department becomes easy. It is also much more transparent to the department that uses the shared service.

There is an easier way to deny access to all objects in an Object Store without using marking sets, which is using a deny of the *connect to store* permission on the Object Store to anyone who must not have access to any object in the store. If wanted, additional to allow only read access, the deny of the *modify existing objects* or *delete objects* permissions might be applicable. This option has the added benefit of not requiring any change to the metadata model, as required with marking sets.

8.5 Gradual security requirement changes

Organizations are fluid environments. Staff and organizational changes are common. Documents might need to be restricted or made available to various audiences. Over time, business processes are modified, types of work are added, and content life cycles might change. There is a need to respond to the changing security landscape while ensuring that any existing information remains protected.

In this section, various issues that are related to a gradually changing security environment are described. It discusses how the IBM FileNet P8 Platform can be used to minimize the administration effort.

8.5.1 How document and process life cycle affects security

As discussed in chapter 8.4.2, “Security policies” on page 234, changes in documents during their life cycles can have implications for those who can access the documents. There is also the question of how to manage the assignment and removal of dynamic permissions. As an example, a customer sends a request to open a bank account. This request and related customer information is not made publicly available to anyone in the organization. At certain point, an account manager is assigned to the case. The account manager then gets the required permissions to the request and customer information to process the case.

In the examples provided so far in this chapter, this use case can be covered by assigning an application security policy template for a state of Account Manager assigned. However, the business requirement might want only individuals who are acting in a particular role to have Add access to the content.

Business process management is effective at retrieving and acting on information in real time. For a particular type of account, or the workload given to particular users, it can automatically be decided which user to assign as an account manager. A supervisory user can also be assigned within the Account Opening process to select the appropriate person to be the account manager. In either case, the information that concerns the person who must be assigned is present as part of the process, and it is the process that needs to act upon it. No document event happened to necessitate this change. It is purely a function of a management process reaching a particular stage.

There are several ways to secure this information within a business process management environment such as that provided by the IBM FileNet P8 Platform (specifically, IBM FileNet Business Process Manager). The user interface restricts who can see a work item. The concept of process attachments are used within a process and security on the queues to restrict who can view information. Although these are valid methods, to be totally sure that content access is secured, it must be ensured that the underlying security permissions are set, which is the only way to absolutely ensure that a user without permission cannot see the content. This feat is possible within IBM FileNet P8 because all expansion products are built upon the Content and Process Engines and are restricted from what they can see and process by virtue of those engines enforcing their security models onto the client application.

This ability brings up a range of questions. How can queues and individual work items locked down? How can the activities dynamically interact with the Content Engine to update security in real time? How are the updates or changes to such activities managed? How are such activities audited? What are the drawbacks? These questions are answered in the remainder of this section.

8.5.2 Managing security updates

To manage the security of an individual document, or class of documents, is relatively simple thanks to the rich features that are available in IBM FileNet Content Manager. After thinking at a more abstract level, things might become more difficult to manage, for instance, when 300 client applications must be worked on in a given week. If these applications are needed to be retained for five years and 100 new ones arrive per week, an additional historical set of 26,000 must be managed.

Let us assume security settings are assigned to the documents when they are being modified or versioned. If the security policy is changed within the organization to reflect a change in business, only the single security policy needs to be modified. Therefore, if the security policies are changed within the organization to reflect a change in how business is done, only the single security policy needs to be modified. Because there are 300 documents in process, all of these documents and new documents in the system have the policy applied the next time they are versioned. If the 26,000 historical records are relatively static, there is an administrative challenge to handle because the security on these are not updated to reflect the new policy.

Conversely, if marking sets are used instead to manage this security, changing this is immediately reflected by all documents within the system. However, the historical documents' security no longer reflects the security that was originally assigned to them at the time of processing, which can be a problem when trying to prove to a regulator that the security on a document, during or at the end of a process, was used to manage it.

Another problem with updating marking sets is that if a marking is removed or added, it is not reflected on any documents that currently use the marking set. Removing a marking from a document because it was removed from a set does not make sense because it can leave the content open. As such, the marking still applies to a document in the Content Engine until that the document is versioned or the property is modified, which still leaves a management overhead with lots of markings to add and remove, rather than just modifying the underlying permissions that they assign to content.

Table 8-8 on page 253 describes the various types of security, the ease of modifying their permissions and adding and removing elements, and therefore the longevity for which must be considered using them in a production, long-lived system.

Table 8-8 Advantages and disadvantages of various security methods

Security Method	Ease of updating permissions	Ease of adding or removing elements	Relative longevity
Direct assignment	Easy for individual documents, but difficult for large groups of documents.	Easy.	Short term. Might require changing many times during life cycle. Useful for short term, dynamic assignments of permissions.
Dynamic Security Inheritance Objects	Easy for large or small groups of documents. Instantly applied to all content. Requires quite detailed system knowledge. Can be used across an IBM FileNet P8 Domain. Segmenting of similar secured objects to assign differing ACLs will be complex.	Easy. An element cannot be deleted if it is in use, but it can be hidden from being selected to prevent its use in the future.	Short-to-long term. Can be used effectively by business processes for application-specific security settings.
Marking Sets	Easy. Instantly applied to all content. Can be used across an IBM FileNet P8 Domain. Segmenting of similar secured objects to assign differing ACLs will be complex.	Difficult because marking sets can only be defined when creating the property.	Long term. Use for rarely changing, enterprise-wide security types where permissions might change but the number of types of access do not change often. Use multiple marking sets if longevity varies considerably.
Security policies / document lifecycle policies	Medium. Easy to change, but is not applied until content is versioned or actioned in a custom application, for example, its state changes. Can be assigned at instant creation time.	Easy for version states, but difficult for a custom application because it requires coding in the application.	Medium. Useful while the document is in use because it abstracts the individual permissions from the document. Not good to lock down content over long periods of time.

Security Method	Ease of updating permissions	Ease of adding or removing elements	Relative longevity
Security folder	Easy and instant thanks to inheritance.	Medium. If a document is filed in multiple locations, users can get confused as to why security for one folder is not being applied. Also, there is no simple interface in Workplace XT to assign a security folder.	Short-to-medium. Useful while document and folder are in use, but difficult to maintain over time for retention purposes.
Default instance security	Medium because it is an administrative task. Only applies to new documents.	Not applicable. This applies to one per class.	Short. Only used at document creation. Useful to initialize owner, policies, dynamic security inheritance objects, and marking set values.

Table 8-8 on page 253 shows that all methods of assigning security to content can be useful but only applicable to specific use cases. To build a comprehensive solution to a problem that requires content to be managed over time, a mixture of methods often must be used.

The remainder of this chapter discusses how to perform changes over time. The example at the end of the chapter pulls these short-to-long term methods together in a real-world example to illustrate how a platform approach to these security issues can assist the management of information security.

8.5.3 Updates using business processes

It is often a requirement to update a document's security over the course of its life cycle. The examples that are provided so far are either manual or controlled by document lifecycle or security policies. There are other scenarios, however, where security might need to be updated dynamically in response to user decisions or information from an external system.

The Process Engine (offered through IBM FileNet Business Process Manager) can be used to execute component steps that call Content Engine API functionality to update security on Content Engine objects. An example of this is when a new customer requests a product and the request must be assigned to a dedicated Account Manager with the necessary permission to manage and see the documents. The Sales Administrator who is in charge of signing the Account Manager, does not need to remember what permissions in the Content Engine to assign to the Account Manager. The Sales Administrator simply selects the appropriate manager and let the business process handles the rest.

The readily available CE_Operations component does not have any security querying or modification functions, but they are simple to create. Consider Example 8-1.

Example 8-1 Sample code to add grantee level

```
public void addGranteeLevel(VWAttachment attachment,String
username,String level) throws Exception {
    Document doc = findDocument(attachment); // utility function

    // create a new access permission object
    AccessPermission ap = Factory.AccessPermission.createInstance();
    Integer l = getAccessPermission(level); // utility function

    // set access permissions
    ap.set_GranteeName(username);
    ap.set_InheritableDepth(new Integer(0));
    ap.set_AccessType(AccessType.ALLOW);
    ap.set_AccessMask(l);

    AccessPermissionList apl = doc.get_Permissions();

    // add the permissions to the list
    apl.add(ap);

    doc.save(RefreshMode.NO_REFRESH);
}
```

Example 8-1 uses the Content Manager Java API to find a document, create a new Access Control Entry, and save the changes. The access mask is computed by adding together the integer value of all permissions to be given to the grantee (user or group). A utility function is used in Example 8-1, so process developers can simply supply *View Content* to specify the level rather than remember the correct integer value for each of the constituent permissions.

Now that the access is secured to the documents throughout a process, it must be ensured that each step is accessed by the appropriate person. It has already been mentioned that queues and rosters can be restricted to particular users, but with IBM FileNet P8, work can be assigned to specific users and workflow groups. A *workflow group* is an array containing references to users in the system. A workflow group is completely different from an LDAP group and is only relevant to a particular process instance. Assigning a step to a workflow group means that each of these users are sent the step to work on. After these users all complete the work step, the routing condition is interpreted as usual.

It is possible to dynamically assign strings that contain user names to elements within a workflow group, which means that for the example the assigned account manager string field can be mapped to a workflow group and then the process can be routed to a step that is linked to this workflow group. In other words, even though at process design time there was no idea who would be assigned to do this work, it can still be effectively designed and the process can be routed. This can happen while guaranteeing that this information is provided for each process instance by a previously assigned user, which makes organizational changes easy to separate from business process design, thus maximizing the flexibility of the IBM FileNet P8 Platform.

8.5.4 Institutional reorganizations

People who move internally or join and leave an organization can prove to be a real security management challenge. This challenge is especially true of systems that assign security to information that is based on an individual rather than just a role. Take the example of an Account Manager who might have document instance security set such that only they can see their customers' documents. If this person leaves, how is that security updated?

> **Avoid mistakes:** Substantial mistakes made by designing the security architecture will cause nameable efforts in adjusting the system and can cause heavy load and long runtimes at the database!

There is no simple answer to this question. Depending on the necessary tasks, there are several approaches that might only be resolved by custom code. In this chapter, some security architecture aspects are discussed; however, for coding issues refer to the appropriate development documentation at the Information Centre.

Some aspects in handling changing security needs are:

- Avoiding dedicated user authorization

  When creating the technical and functional security concept, it is important to avoid any dedicated user authorization. Will it be necessary in spite of precautions to define dedicated user authorization? It is essential always to define at least one additional group-based ACE to keep a valid access right to the particular objects as an administrative fall back.

Additional object protection—if necessary and in addition to others—can be defined using marking sets or only temporary memberships in the fall back group.

Prevent arbitrary individual security configuration.

- Controlling individual document entry

  IBM FileNet P8 offers a bundle of methods for controlling individual document entry:

  – The Default Instance Security, especially the #CREATOR-OWNER ACEs, can be used for dedicated object classes for object entry to the P8 environment for individual purposes (for example, if the Content Engine is used as a compliant personal document repository alongside enterprise-wide processes).

  – Document Entry templates can preconfigure security, as needed, to meet individual, business, and administrative needs.

- Using properties for granular object grouping

  Using properties for grouping objects supports various possibilities to reorganize objects by means, such as executing search queries and related actions on the result list. Remember that there is no query that allows defining a search based on ACEs on Content Engine objects.

  Using more than one property for object grouping purposes can enable (within limitations) one-to-many reorganizations.

- The *Process Administrator Tool* can be used to identify open work items for the dedicated user (for example, Work Item search query for *F_BoundUser*) and appropriate reactions to complete open work steps are decided by the responsible user in the *Process Administrator Role*. Changes to the object securities must not be made before impact to open work items are discussed.

- While designing a workflow map, it might be necessary to relate some steps to dedicated users (without standard redundancy) instead of groups. In this case, it is necessary to wrap that step in an escalation timer that escalates the task after a period of time by reassigning the step. This design mitigates personnel changes and also absence on vacation or other long-lasting reasons.

  To prevent assigning new work to absent users without knowing the individual replacement person, use the out-of-office functionality of the Process Engine. Any work assigned to the user is assigned to the designated proxy user. The definition of the proxy user can be done by the given user to be covered using the user's **Preferences** → **Tasks** → **Out of Office** configuration within the Workplace XT. A centralized way for the Out of Office configuration is using clicking **Process Administrator Tool** → **Tasks** → **Out of Office***.* The responsible user in the Process Administrator Role can select a user of interest and define the proxy user.

8.6 Content-level security

After the system verifies that a user is actually authorized to access a piece of content, be it properties on a document, the content itself or a workflow item, security must also be ensured wherever that information flows. In this section, methods are described that are currently in use that can be taken advantage of. It is strongly advised that anyone who is interested in this section also seek the advice of a *FileNet Certified Professional* or *Solutions Architect* to discuss specific security requirements. There is a plethora of internal security features, such as encryption and storage of security keys and credentials, that are not mentioned in this section, but that might be of major interest in a given situation or requirement.

8.6.1 Local copies on user machines and client cache files

A major problem for organizations is preventing duplicate, local copies of content on individuals' machines. This problem is especially important when introducing a new enterprise content management system because there is usually some resistance to changing working practices by storing documents in a central system rather than keeping personal copies. The quicker this is resolved, the easier ensuring compliance becomes.

Various user access methods are possible with the IBM FileNet P8 Platform, which includes thin applications, such as Workplace and Business Process Framework, their viewers such as the Image Viewer, or integrated desktop applications, such as Microsoft Office.

Whenever a browser accesses a web page or download, the application makes a temporary local copy. It is advisable, therefore, that if the security of information is paramount, consider using controlled access methods rather than allowing general download access for content.

The application integration capabilities of Workplace and Workplace XT can be used to bring content under control while providing a rich and intuitive user experience to encourage adoption of the new system. Application integration embeds a set of IBM FileNet P8 menu options in Microsoft Office applications to open, check out, and check in documents from an IBM FileNet P8 repository.

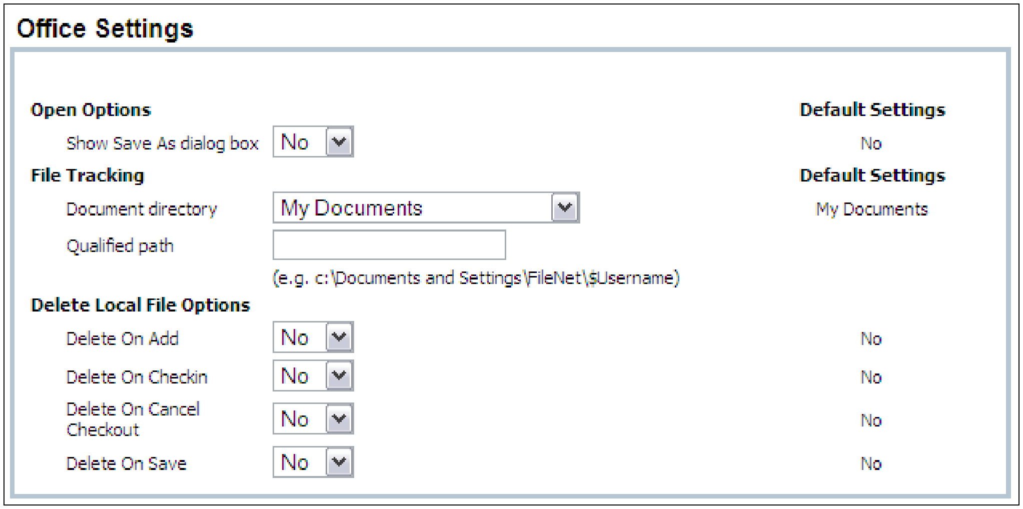
If using Application Integration, FileNet Integration for Microsoft Office, or Workplace XT, there is an embedded application called File Tracker.

Note: Workplace and Workplace XT provide various File Tracker applications that cannot coexist on the same PC.

It monitors all documents that are downloaded using Workplace or Workplace XT and Office Integration and provides extra usability by tracking the repository object that a local file represents. This tracking makes checking in modifications to content fast because the system does not have to ask the user which document to check in.

The File Tracker has some extra, centrally configured settings that allow administrators to specify when local copies of content are deleted, which are configured in the Workplace or Workplace XT Site Preferences page, as shown in Figure 8-20. The most common scenario is to remove a local copy of a document when it is checked in to the content repository. This action ensures that there is always only one current version of a document within the organization.

Remember that the files, as long as they are checked out and stored locally (and that might be an unprotected network share as well), are not protected against improper access and changes.



Office Settings

Open Options		**Default Settings**
Show Save As dialog box	No	No
File Tracking		**Default Settings**
Document directory	My Documents	My Documents
Qualified path	(e.g. c:\Documents and Settings\FileNet\$Username)	
Delete Local File Options		
Delete On Add	No	No
Delete On Checkin	No	No
Delete On Cancel Checkout	No	No
Delete On Save	No	No

Figure 8-20 File tracking options to ensure local copies are deleted

8.7 Network security

There are many issues to deal with when accessing an application over a network, especially if that network is potentially insecure. In this section, the issues are detailed and how they can be mitigated, thanks to the features of the IBM FileNet P8 Platform.

8.7.1 Demilitarized Zones

The classic approach to network security is to have a secured server layer that only allows access from specific machines in a so called Demilitarized Zone (DMZ). The DMZ is a protected area (usually by firewalls) with limited communication possibilities (protocols, ports, allowed IP-addresses, and so on) between the systems within and outside of the DMZ. One or more additional zones, with no direct communication allowed to systems, can exist within the DMZ.

The IBM FileNet P8 Platform supports this methodology and can be configured in several ways. A simple solution is a departmental system without any high-availability requirements and only a single zone within the DMZ. Workplace XT and Content engine might even use the same server. Access is made directly by a client using a browser or Microsoft Office client, which allows them to flow through the Workplace XT Web application. This application can be installed in the DMZ with the Content Engine, Process Engine, and database servers located in the same server layer, as shown in Figure 8-21 on page 261.

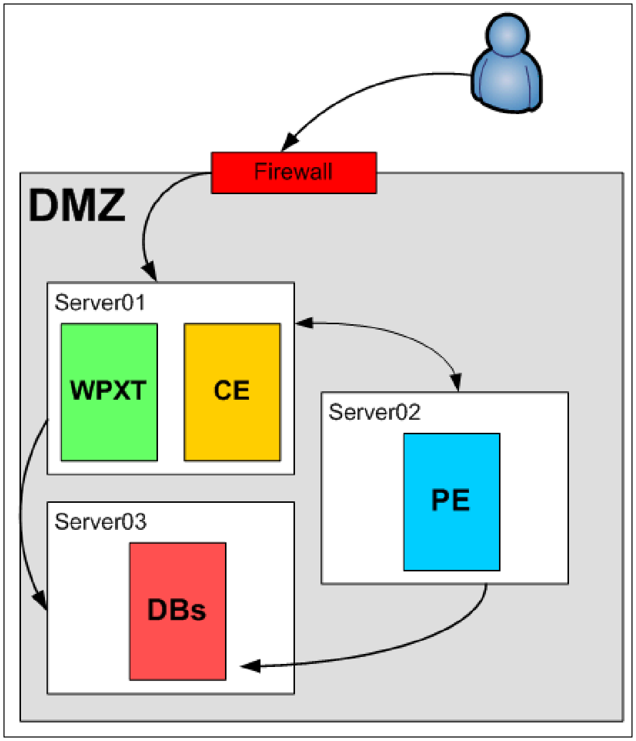


Figure 8-21 General illustration of a simple DMZ

On the opposite end of the scale, one might have a highly-available environment with clustering that is protected by secure protocols (for example, https, SSL off loading on hardware load balancers, or software based SSL off loading discussed in the next chapter) and secure web proxy products. The DMZ in Figure 8-22 on page 262 is divided into three zones in which access is restricted from one level to the direct following.

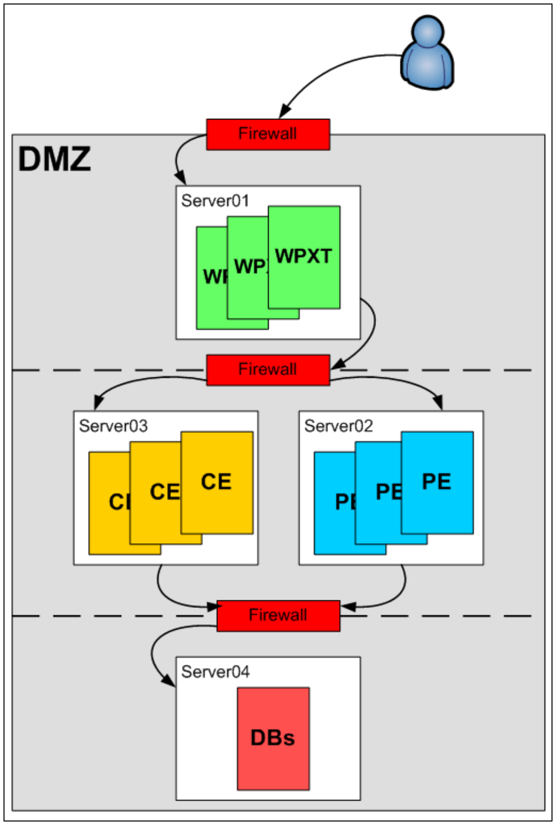


Figure 8-22 General illustration of a complex DMZ

The advantage in Figure 8-22 is that only the secure proxy is exposed in the DMZ. This method is used across many applications and not just IBM FileNet P8 in typical implementations. As such, there are fewer opportunities for compromising security in the DMZ layer because all application servers are located in the server layer. Having a highly-available environment also means that clients connect through one IP address that points to a (typically hardware) load balancer, which makes configuration of firewall rules easier to manage.

In the server layer, itself, the Workplace XT, Content, and Process Engines communicate to each other over known ports and protocol.

8.7.2 Encryption on the wire

It is often desirable to ensure that all clients are accessing content through a secured channel. Inside the DMZ and in the server layer(s) are defined to be confident that network traffic is protected. On a network with a large number of potentially compromised computers, however, extra steps are necessary to prevent unauthorized access to content.

All IBM FileNet P8 Platform web user interfaces are supported in application servers that use the Secure Sockets Layer (SSL) technology, which is desirable in non-Single Sign-On (SSO) environments where users enter their user name and password directly into a web page and in SSO environments to protect the communication itself. Many high-profile security breaches occur in situations where the login page of a site was not on a page that was protected by SSL and the target service was. It is a good idea to ensure that SSL is always used alongside the entire communication way from the client application to the hardware balancer immediate before or at the application server that hosts the Workplace XT.

SSL can be configured to provide three important data security features:

- The initiating handshake is used to identify the client that is sending the request.
- SSL can be configured to validate that the data was not modified in transit.
- SSL provides data confidentiality by encrypting data over the wire.

Behind the scenes in the server layer, it might also be wanted for all elements of the IBM FileNet P8 Platform to communicate securely with each other. This communication includes authorization lookups to the underlying Directory Server through LDAP and interaction between the core engines, such as Workplace XT and the Content Engine. SSL can be used to encrypt these internal communications.

It is important to note that some communications used by the IBM FileNet P8 Platform rely on the security support of the underlying systems. The JDBC database driver, for example, must support secure communications with the chosen database vendor's product; likewise, the application server's JAAS login module requires configuring to perform authentication requests using secure channels. While this is out of scope of the IBM FileNet P8 Platform, it is worth mentioning that to ensure as secure a system as possible, security all the way down the application stack into the database and storage layers is also an important, and often overlooked, consideration.

Note: Depending on the secure protocol that is used for client/server communication, issues concerning hardware load balancing might exist. Likewise there are dependencies between using firewalls, used protocols, and application-server products. It is absolutely necessary to contact the manufacturer of the selected hardware and software products and refer to the latest documentation at an early stage of the design phase.

8.7.3 Web services security

All web services APIs for both the Content Engine and Process Engine are supported over an SSL protected connection. By default, all information that is known to be sensitive, such as user names and passwords that are sent to the Content Engine, are encrypted using asymmetric public private key pairs. These keys are generated during installation of the Content Engine. For more information, see the *IBM FileNet P8 Platform Installation Guide* at:

https://www-304.ibm.com/support/docview.wss?uid=swg27010422

The Content Engine EJB transport options, which include RMI-IIOP and T3 (Weblogic), both support the use of SSL to protect communication. Communication through the Web services APIs can also be secured using SSL-protected HTTP (https). It is important to identify limitations based on the actual planned architecture and the planned software to be used.

The Process Engine Java API connects to the Process Engine directly using RMI-IIOP, and for this API, the channel does not support SSL-protected communications. Network-based encryption techniques that preserve the IP packet must be used if this communication must be protected. Because all login and document retrieval requests are processed through the Content Engine Java API, this is only an issue if some of the process fields contain sensitive information because session tokens are transmitted encrypted.

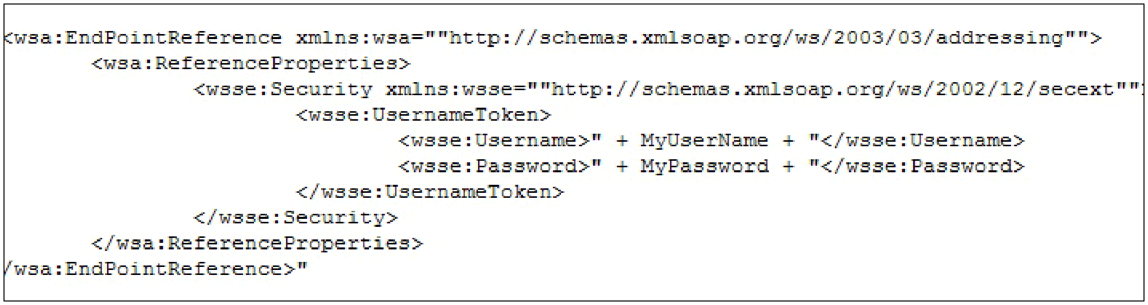
IBM FileNet P8 processes can invoke, receive, and reply to Web services calls. The security aspects of these processes are important to discuss. The standards supported by the Process Engine's support for web services are:

- WS-BPEL for managing the orchestration conversations with services
- WS-Security for passing authentication and authorization information to those services

IBM FileNet P8 can interact with any web services that are protected using SSL communications. This communication is handled by the WSRequest component queue that is installed in the client application. The incoming messages are handled by a servlet that is configured in the Workplace or Workplace XT applications, depending on preference. To protect all incoming Web services messages requires configuring at least one Workplace/Workplace XT instance to be SSL protected.

Providing credentials within an IBM FileNet P8 process requires some manipulation of the web services partner link variable. Doing so requires changing the underlying XML header to include WS-Security information to the target web service. This is a flexible method and is the same approach that is used to dynamically change the host target for your web services call, which can be useful in situations where one wants to off load the web services call to a separate geography or service provider.

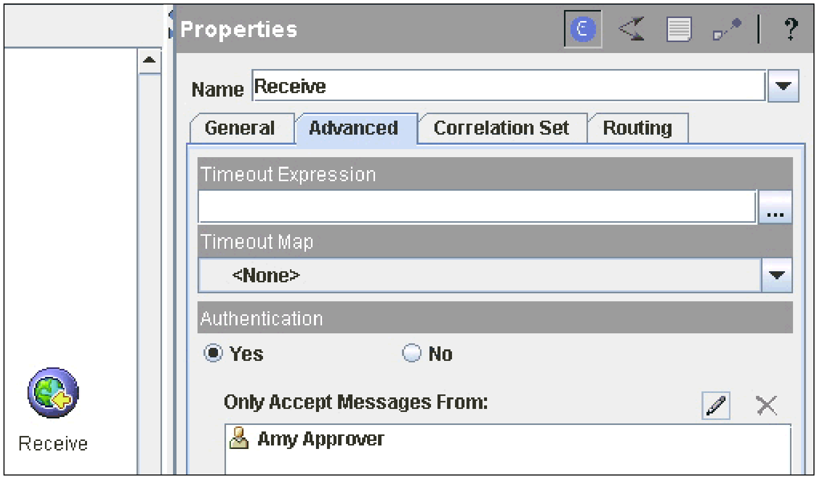
In Figure 8-23, MyUserName and MyPassword are passed in two process fields to generate the web services security header.



```
<wsa:EndPointReference xmlns:wsa=""http://schemas.xmlsoap.org/ws/2003/03/addressing"">
       <wsa:ReferenceProperties>
               <wsse:Security xmlns:wsse=""http://schemas.xmlsoap.org/ws/2002/12/secext""
                       <wsse:UsernameToken>
                               <wsse:Username>" + MyUserName + "</wsse:Username>
                               <wsse:Password>" + MyPassword + "</wsse:Password>
                       </wsse:UsernameToken>
               </wsse:Security>
       </wsa:ReferenceProperties>
/wsa:EndPointReference>"
```

Figure 8-23 Providing WS-Security credentials for a Web service

For incoming web services requests, the process can be instructed to validate the incoming user name and password information against a specified set of user accounts. The Process Engine then checks with the authentication provider that IBM FileNet P8 uses to ensure that the user name and password are valid, as shown in Figure 8-24.



Properties

Name Receive

General | Advanced | Correlation Set | Routing

Timeout Expression

Timeout Map

<None>

Authentication

● Yes ○ No

Only Accept Messages From:

Amy Approver

Figure 8-24 Configuring incoming authentication for a web services call

Security can only be provided to invoke external web services that store the authentication information in a WS-Security header. Some technologies, such as protected .NET 2.0 Web services, do not use this method and instead protect access by performing HTTP header manipulations and handshakes. These methods are not supported by the IBM FileNet P8 Platform. To interact with these services, WS-Security header support must be enabled rather than restrict access to the service through HTTP authenticated handshakes. It is also relatively trivial to create an unprotected .NET 2.0 Web service to accept the incoming request, extract the credential information, and invoke the target-protected Web service using HTTP headers. This works similarly to a proxy for Web services.

8.8 Auditing

In this section, the kinds of logging and auditing that the IBM FileNet P8 Platform supports, its core engines, and the most common expansion products are discussed. A brief summary of information is available both to security professionals and business managers to inform their activities and prove when and why particular actions are taken in the organization. These events range from small, atomic events, such as changing a document property, to major events that occur over the life cycle of a document or the course of a business process.

Auditing can be used to satisfy compliance requirements or maintain vital business interests. Logging on a user basis, to monitor the activities of a dedicated user, is not available when user information be part of the log entry.

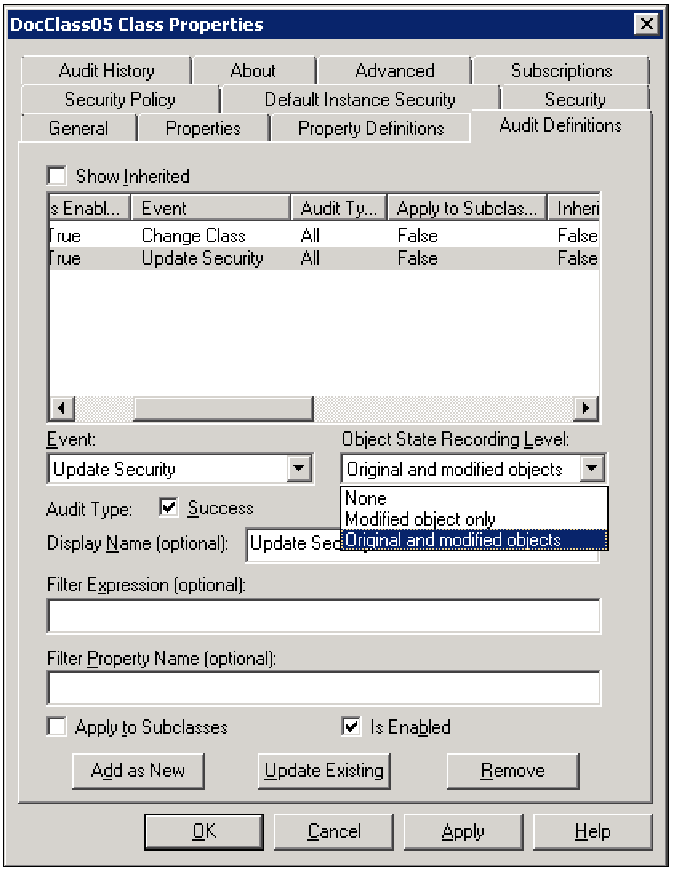
8.8.1 Logging in the Content Engine

As mentioned in Chapter 7, “Building an ECM solution” on page 165, the Content Engine supports auditing of most actions that happen to Content Engine objects, which includes all actions that can occur to a document, custom object, or folder. These events can be subscribed to facilitate *active content* processing and can also be configured on a per-class basis to write audit logs for specific, configured actions on important content. It might be chosen, for example, to log all document check in and delete events in the repository. For certain critical document types, it might be necessary to enable logging of the viewing of properties or content of a document to be absolutely certain about who had access to the content and when (see the note at the end of chapter 8.8.1, “Logging in the Content Engine” on page 266).

To enable auditing on the related Object Store’s properties General tab, the **Auditing enabled?** option must be selected (Default: not selected). In addition, Audit Definitions must be created for the classes and events that are to be audited, as shown in the example in Figure 8-25 on page 267. The audit will immediately be enabled.

The log information is represented as a subclass of the Event object within the Content Engine, which means that it is stored as a row in the underlying object store database. As with any other object, instances of these audit items can be searched for and their properties retrieved. This process occurs through either the standard Content Engine search support or using the read-only JDBC provider. The audit history can also be analyzed at the Audit History tab of a given object with active auditing.

On a security perspective, especially the Update Security event will be of logging interest, as shown in Figure 8-25.



□ Show Inherited

s Enabl...	Event	Audit Ty...	Apply to Subclas...	Inheri
True	Change Class	All	False	False
True	Update Security	All	False	False

Event: Update Security

Object State Recording Level: Original and modified objects

None
Modified object only
Original and modified objects

Audit Type: ☑ Success

Display Name (optional): Update Sec

Filter Expression (optional):

Filter Property Name (optional):

□ Apply to Subclasses ☑ Is Enabled

Figure 8-25 Configuring Security Update logging event

Note: If the object’s initial class will be changed, the initial audit configuration will not affect future objects.

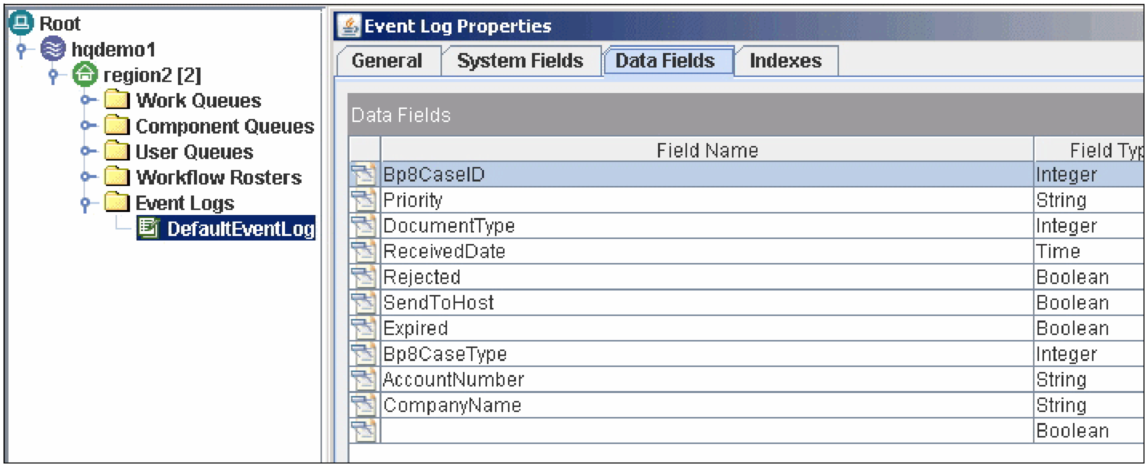
Trace log files can also be created to log these (security) events, lower-level debugging, and diagnostic information for the Content Engine itself. The trace log is configured on the P8 Domain properties Trace Control tab. It is not recommended to use log files for security monitoring because they are not protected by the IBM FileNet P8 system and might be accessed by unauthorized personnel. On the other hand, the audit events can be seen only by personnel that either is allowed to have access rights to searches that might be able to search the audit events or access rights to the audited objects.

> **Note:** Audit-Logging not only allocates space in the database but also creates performance overhead. Keep in mind that data collection must always be kept at the inevitable minimum. The logging of read access (for example, Get Content-event) results in an additional database consumption. In any case, the impact caused by logging must be included in the storage and performance calculation.

8.8.2 Logging in the Process Engine

The Process Engine log is concerned with all life cycle actions that occur during the course of a business process, which includes workflow creation, step completion, and exception handling. These events can be logged into a particular log database table. Multiple logs, each with its own database table, can be configured for the same Process Engine region, which allows the use of one log, for example, for several processes that make up a single application.

Each log that is shown in Figure 8-26 can be configured to record additional user-defined process fields to produce logs with more contextual information.



Data Fields

Field Name	Field Typ
Bp8CaseID	Integer
Priority	String
DocumentType	Integer
ReceivedDate	Time
Rejected	Boolean
SendToHost	Boolean
Expired	Boolean
Bp8CaseType	Integer
AccountNumber	String
CompanyName	String
	Boolean

Figure 8-26 The default Process Engine log with extra configured user-defined fields

The type of Process Engine events to be logged are configured at the region level. These events can be predefined or user-defined events. These user-defined logs can be populated using the *Log step* type within a business process map. The event log to use is passed, the custom log message identifier, an integer, and the message to log. The Process Engine then handles the collection of the additional, configured process fields and logs the entire message in the specified log file.

This information is secured by preventing any user, without access to the corresponding process roster, from accessing any of the log items that correspond to it.

8.9 A practical example

As a practical example, we consider an enterprise content management solution with business process management features, the re-insurance placement and litigation application. Re-insurance encompasses the concept of large organizations that must ensure, for example, five buildings, a fleet of 80 cars, mobile phones, and a variety of other instruments. They use re-insurers as a single point-of-contact for all of their insurance needs. These re-insurance organizations then break down the total requirement into individual parts that they can get other insurance firms to in turn insure. In our example, this situation can mean that all vehicle insurance is grouped together and sent to three companies to bid on that part of the overall requirement.

These bids are then collected, and a full package cost is calculated with terms and conditions. A markup is added to cover costs and to create profit for the re-insurer, and this final amount and terms are presented to the re-insurance customer.

In our example, the flow of information is:

1. Customer submits a large requirements document to the re-insurer.
2. The document is scanned and routed to the account team.
3. The assigned manager receives the information and starts to put together individual elements to solicit bids.
4. These individual forms are submitted to a select few external insurance agents.
5. When all bids come back, or the time to bid is exceeded, they are evaluated by the account manager who chooses a preferred vendor for that element.

6. A total charge amount and legal terms are created and sent to Legal for approval.
7. After they are approved, they are sent to the customer who signs an agreement or asks for a reassessment of part of it or abandons the request.

Designing the business process is out of the scope of this book. Instead, we discuss the security issues that it entails:

- Providing access for the same account team to all documents related to a particular customer.
- Hiding customer documents from other account teams.
- Ensuring that documents do not leave the customer's legal jurisdiction.
- Preventing important competitive information from being available to some people on the account team.
- Creating bid elements by the account manager.
- Tracking the instructions that an account manager sends to insurers, and what we receive back.
- Ensuring that late bids are not considered.
- Preventing bid elements from being modified after the total customer charge and legal terms are considered.
- Recording every quote that we send to a document.
- Recording all agreement documents signed by a customer.
- Managing ad-hoc correspondence.

Providing access for the same account team to all documents related to a particular customer

A customer folder is created with *access permissions* set such that the account team can view properties and content of any document that is linked to this folder by the *security folder* property. All subfolders are set up to inherit permissions from this folder. The parent folder itself, must not be set to inherit permissions from its higher-level folder to prevent unauthorized access.

We ensure that new documents that are being scanned into or added to the system initiate a Content Engine action handler that sets their security folder attribute to the correct value. We find the appropriate folder by making sure that the *customer folder* is a special class of folder with a required *customer number* attribute.

We can optionally create special business processes or subfolders to give a greater granularity to where documents are filed. If we receive hundreds of documents from this customer that relates to different matters, for example, we might want to file the incoming documents into the appropriate subfolders.

Hiding customer documents from other account teams

We must ensure that the *customer folder* does not inherit permissions from its parent folder. To ensure that security cannot be changed, we must remove any *modify permissions* rights from all non-essential users. We can limit this to just the *Account Manager* on the parent folder, for example, but allow no other person to have this right.

We must also ensure that no privileged permissions are assigned to users, which includes the *write any owner* and *privileges write permissions* on the object store. A user with these rights can modify the document owner and through this its security or modify the content, respectively.

We ensure that all customer documents have *null* in the owner field. We use a separate management workflow process to change ownership rather than allow this to occur manually.

Ensuring that documents do not leave the customer's legal jurisdiction

We create a required multi-value property on all customer documents called *applicable jurisdictions*. We create a marking set that consists of markings that restrict access from users in particular locations, which ensures that the document cannot leave the customer's applicable jurisdictions. We deny all rights from any users outside of applicable jurisdictions who use the constraint mask for each marking.

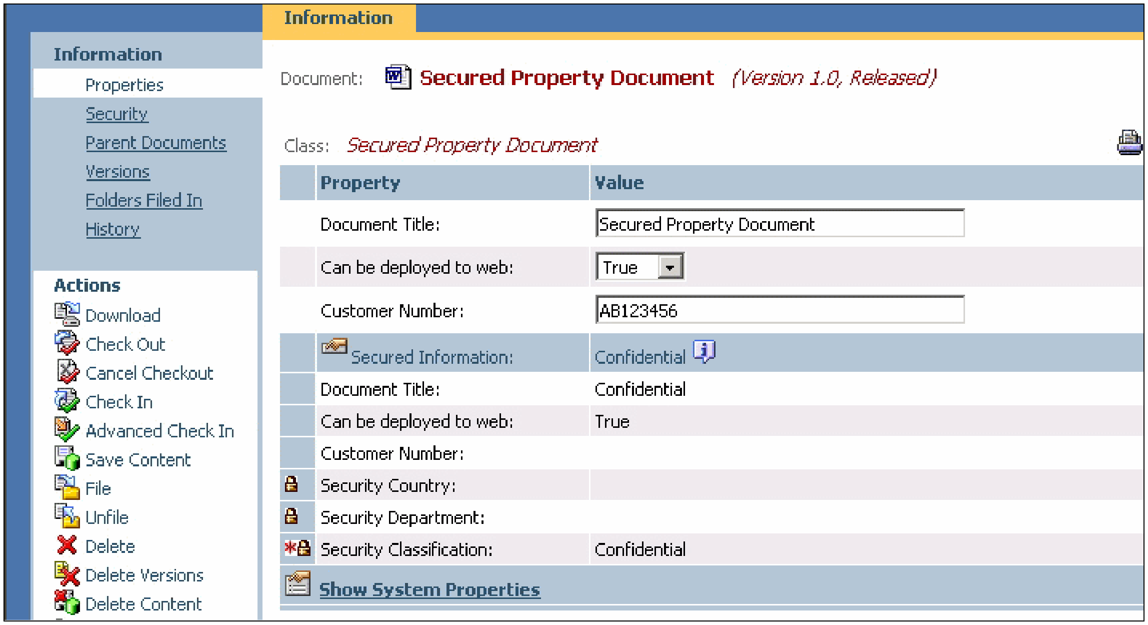
Preventing important competitive information from being available to some people on the account team

Some document properties, in addition to document content, might include competitive information, such as the value of a quote. We might have some people who are on the account team, such as customer service representatives, who must not see this sensitive information.

At the same time, we might have properties on the document that these users should see, which can be so that they can confirm to a customer that a document was indeed received.

We cannot directly secure individual properties on a Content Engine object, but we can secure objects themselves. Thus we can create an object to hold our securable properties and link this to the document by using an object value property. This object can be set up with a *confidential* security marking to deny the user from seeing this information.

We create a class called *secured property document*, which contains a *secured information* object value property that is required to point to one or zero *security showcase document* objects. These security showcase documents are required to have a value that is set for their security classification. As you can see in Figure 8-27, logging in as a user with sufficient clearance means that we can see the object and its metadata.



Information

Information
- Properties
- Security
- Parent Documents
- Versions
- Folders Filed In
- History

Actions
- Download
- Check Out
- Cancel Checkout
- Check In
- Advanced Check In
- Save Content
- File
- Unfile
- Delete
- Delete Versions
- Delete Content

Document: Secured Property Document (Version 1.0, Released)

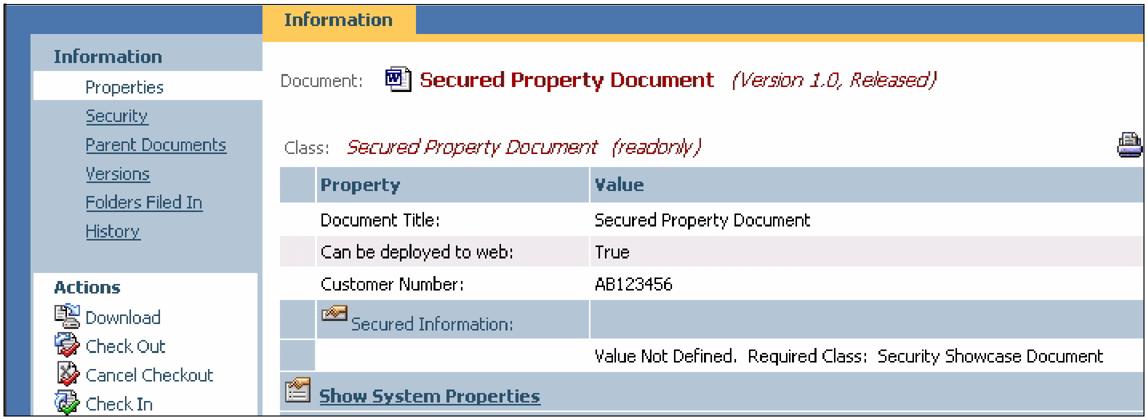
Class: Secured Property Document

	Property	Value
	Document Title:	Secured Property Document
	Can be deployed to web:	True
	Customer Number:	AB123456
	Secured Information:	Confidential
	Document Title:	Confidential
	Can be deployed to web:	True
	Customer Number:	
🔒	Security Country:	
🔒	Security Department:	
*🔒	Security Classification:	Confidential

Show System Properties

Figure 8-27 Logged in as Administrator, we can see the confidential properties

If we instead login as a user without confidential access, we can read other document properties, but we cannot see the confidential object or its properties, as shown in Figure 8-28 on page 273.



Document: Secured Property Document (Version 1.0, Released)

Class: Secured Property Document (readonly)

Property	Value
Document Title:	Secured Property Document
Can be deployed to web:	True
Customer Number:	AB123456
Secured Information:	
	Value Not Defined. Required Class: Security Showcase Document

Show System Properties

Figure 8-28 Logged in as a user who cannot see the confidential information

We also can define this property as having the document's security inherit from the secured information property and make this property required. If we then set permissions on the confidential information such that view content was denied and inherited by the confidential document and immediate children, and view properties was denied for the confidential document only, there is no need to explicitly deny view content rights to the top-level document.

In other words, the view content rights are determined by how sensitive the secured information was. In practice, this is useful if the properties are derived from the main document's content, which means that this security setting can be applied one time on the secured information object and not have to also add a permission to the main document to deny just view content to specific user groups. If confidentiality changed over time, this option is desirable from a manageability perspective.

Note: Do not forget that an explicit direct allow permission on the document overrides an inherited deny for *view content permissions* in this case, which illustrates the desirability of using inherited permissions over direct or default allow permissions.

Creating bid elements by the account manager

We use a business process to create a skeleton structure for a customer deal, which includes a bid elements folder. The Account Manager creates subfolders of class bid element that contains a required bid identifier property. We give the account manager create subfolder rights on the bid elements folder. We also ensure that, by default, the #CREATOR-OWNER of a bid element folder has the file in folder permission.

Because all subfolders of the top level customer folder inherit security permissions and all documents have their security parent property set, we ensure that anyone on the account team can also view properties and content of documents in the bid element folders, which means that the account manager does not have to worry about administering security and can instead concentrate on working on the content.

Tracking the instructions that an account manager sends to insurers, and what we receive back

We require an account manager to complete an electronic form with an instruction that is sent to all insurers who are asked to bid on an element. This form drives a business process. The bid element folder is attached to the business process at the same time the form is completed by using a form workflow policy process.

This policy locks down the form so that the account manager cannot modify its content to change what the instruction says or who the bid request is sent to after the form is submitted. To ensure security at the account manager's workstation, we require that the electronic form have a digital signature that the account manager signs just prior to submission. We make this lock down the fields on the form to prevent modification of a signed form by any privileged user.

The business process collects the relevant bid element documents and submits them using an encrypted Web services call to the relevant insurers. Any responses from the insurers are authenticated with the relevant user name and password for this insurance organization to ensure that fake bids do not enter the organization.

Ensuring that late bids are not considered

Upon expiration of the bid timer, the business process locks down the quotes subfolder for the relevant bid element, which prevents an *incoming bid process* from being able to add a bid after the timer is exceeded. If the bid comes later, the incoming bid process can reply to the insurer, notifying them automatically that their bid is not accepted because the submission deadline is not met. It also means that the account team does not worry about whether or not a bid is late. For large accounts, if doing it manually, late bids can be mistakenly considered or modified after submission. By making this electronic and secured, we can prevent this from becoming a problem.

Wait for condition: You can model a *wait for condition* in an IBM FileNet P8 business process to wait for a known maximum number of launched subprocesses, for example, you can have conditions on five routes to launch or wait for a maximum of five sub processes. Use a custom object with an array field called, for example, *SubProcessStates*. In the IBM FileNet P8 process, you launch each sub process in a loop passing it as reference to the custom object instance and the sub process' own index number. Each sub process is responsible for updating its status within this custom object. The main process then periodically checks these values to determine when to continue.

An alternative to periodic checking is to have a Content Engine Java event handler check the custom object every time it is updated. If it finds that all status entries are marked as complete, it can fire a single re-awaken process. The main process then has a wait for condition waiting for this single process, which puts the minimum amount of load on the Content Engine and Process Engine for all of the process synchronization because the wait for condition values are checked when the waited-for process is modified.

Preventing bid elements from being modified after the total customer charge and legal terms are considered

In the re-insurance handling process, we have a system step to add an explicit deny permission for property and content modification to every bid element folder. Because no one can manually add a permission to a bid element document (thanks to default instance security), we know that the modification deny permission at the folder level cannot be overridden by a direct permission on the bid element document because no one can manually add any such permission.

Recording every quote that we send to a document

In case of legal action, we declare all documents that are involved in a re-insurance quote as a critical business record just prior to sending a quote response to a customer. We can use IBM Enterprise Records to manage the access to this information so that applying legal holds and managing disposition is automated and centrally controlled according to organization-wide policies, defined by the Legal and the records management teams.

Recording all agreement documents signed by a customer

When a customer agrees with our quote, we must also lock that information down and declare that as part of the same business record. Again, we use IBM Enterprise Records to do this.

Managing ad-hoc correspondence

We use the email management features of the IBM FileNet Content Collector to capture all customer correspondence and put them in the relevant correspondence folder. The correspondence is located in our customer folder and is automatically declared as a business record. We can, for example, have a default email storage policy of three years for all customer correspondence. The email can also be declared as a re-insurance quote acceptance record if it is related to a customer re-insurance request, perhaps with a different retention period. IBM Enterprise Records resolves these two record types and retains the correspondence accordingly. If bid-related documents are held for five years, for example, this prevents the disposition of the email after the usual minimum of three years, which allows us to prove compliance across different sources of content.

8.10 Summary

This chapter showed that there are many factors to consider when implementing an IBM FileNet P8 solution, which encompasses all elements of security, from authentication and authorization of users to encryption of communications, storage of content, through to proving compliance. Auditing and proving who has or has not modified content is just as important as securing information, and this is increasingly true due to increasing regulation and the litigation nature of business in the current environment.

The IBM FileNet P8 Platform has a rich set of security and auditing features. These features can be used by expansion products, such as IBM FileNet Business Process Framework, to provide the same capabilities to new business solutions. Auditing and security management showed how they can be extended with other IBM FileNet P8 Platform expansion products, such as IBM Enterprise Records.

The IBM FileNet P8 Platform provides a comprehensive set of security features that are sophisticated enough to solve a wide range of business and IT problems. This chapter explained how these features can be applied to new, complex business problems to enable deployments to be conceptualized quickly, which enables customers to rapidly configure, adapt, and extend an IBM FileNet P8 Platform-based solution, while maintaining tight control over the security of the overall system.

9

Infrastructure and scalability

This chapter describes the infrastructure used with IBM FileNet P8 and how a solution can be scaled to respond to increasing demand. Also discussed is how a hub and spoke architecture can extend IBM FileNet P8 to satellite offices or support a world-wide distribution.

This chapter covers the following topics:

Disclaimer: This chapter is not an exhaustive exploration of every possible install scenario. Consult a certified IBM FileNet professional when planning a production architecture.

9.1 Overview

IBM FIleNet P8 is a flexible platform supporting multiple operating systems, databases, user directories, and storage technologies. This gives organizations the freedom to choose the supporting products which fit within their IT strategy. This section explores the infrastructure of a FileNet solution and how it can be scaled to meet user demand.

Table 9-1 lists some of the terms used throughout this chapter.

Table 9-1 Terms and acronyms used in this chapter

Term	Definition
DMZ	Demilitarized zone An area that is normally enclosed by two firewalls to secure internal servers from any harm that originates from potentially insecure external networks that need access to resources in the internal servers.
CORBA	Common Object Request Broker Architecture An architecture that the Object Management Group (OMG) manages that allows implementing distributed applications.
RMI-IIOP	Remote Method Invocation over Internet Inter ORB Protocol A communication method that CORBA frequently uses.
JAAS	Java Authentication and Authorization Service A standard to implement authentication and authorization in the context of a J2EE application.
Content Engine or Process Engine API	The application programming interface that the respective IBM FileNet P8 engines provide.
EJB transport	Enterprise Java Bean transport Communication mechanism that the Content Engine clients use to directly communicate with the Content Engine server EJBs.
WSI transport	Web service interface transport Communication mechanism that the Content Engine clients use to communicate with the Content Engine server through the web services listener.
WORM	Write Once Read Many Ensures that data written one time cannot be altered afterwards.

9.2 Supporting technologies

The IBM FileNet P8 software platform is not intended to run as an independent application. Instead the software is designed to be open and integrate well with pre-existing infrastructure. An IBM FileNet P8 solution is supported by several technologies:

- A database, such as IBM DB2
- An LDAP directory server such as IBM Tivoli Directory Server
- An application server such IBM WebSphere Application Server
- A storage platform such as IBM System Storage® N-Series
- Optionally an archival platform such as IBM System Storage N-Series Snaplock or IBM Information Archive
- In most cases a hardware load balancer such F5 Networks BIG-IP Local Traffic Manager or Citrix Netscaler

These products contribute to the ability to scale P8.

9.3 Horizontal versus vertical scalability

Organizations using IBM FileNet P8 need the ability to match system performance with application demand. Demand for an ECM application is driven by organic factors such as increasing adoption by existing users or growth of the organization itself. Growth is also driven by new applications and work functions migrating to P8.

IBM FileNet P8 capacity can be increased by adding servers or moving to a larger server. The larger server method is known scaling vertically whereas the add server method is known as scaling horizontally. Historically speaking, the vertical method was relied on almost exclusively. To scale vertically an application is migrated from a smaller server to a larger server which can accommodate additional processors, memory and other components to improve performance. The drawback to this approach is the expense of migrating the application.

The vertical methodology remained popular until the network industry began to develop specialized switching hardware which can distribute user traffic to multiple systems, As a result, more recently developed applications support horizontally scaling where servers running like software share the workload. The drawback to the horizontal method is the complexity of the design (in this example, stovepipe or bowtie design methodologies), lack of universal adoption, the number of horizontal scaling technologies, and the presumption that the capacity to server relationship scales linearly.

Horizontal scaling is becoming the standard for enterprise applications, but software not initially designed to scale horizontally will usually scale in a vertical fashion only. Table 9-2 is a typical IBM FIleNet BPM stack running IBM Enterprise Records and how each component is able to scale.

Table 9-2 Scaling Methodologies

	Horizontal scaling	Vertical scaling
Scale using hardware only	IBM DB2 for z/OS (using Sysplex) Workplace / Workplace XT (using load balancer) Process Engine (using load balancer) ECM Widgets (through load balancer) IBM Enterprise Records (through load balancer) Content Engine (WSI using load balancer) IBM Tivoli Directory Server (using load balancer) IBM System Storage N-Series	► IBM DB2 for z/OS ► IBM DB2 for Midrange ► Workplace / Workplace XT ► Process Engine ► Content Engine ► Content Search Services ► Rendition Engine ► Case Analyzer ► Process Simulator ► ECM Widgets ► IBM Enterprise Records ► IBM Content Collector ► IBM Tivoli Directory Server
Scale using software only	IBM DB2 for Midrange (using pureScale™) Workplace / Workplace XT (using IBM HTTP Server) ECM Widgets (using IBM HTTP Server) IBM Enterprise Records through IBM HTTP Server) Content Engine (WSI using IBM HTTP Server) Content Engine (IIOP using WebSphere) IBM FileNet Content Search Services (using product feature) IBM Content Collector (using product feature)	N/A
Scale using software and hardware in tandem	Workplace / Workplace XT IBM Enterprise Content Management Widgets IBM Enterprise Records IBM FileNet Content Engine (WSI only)	N/A

9.3.1 Horizontal scaling: Scale out

Horizontal scaling, also referred to as *scaling out*, is based upon the principle that the performance and throughput of the total system can be increased by adding *physical servers.*

Depending on the internal architecture of the application, it might be possible to implement horizontal scaling just by installing the appropriate software components on the additional machine and configuring the application to use them. Commonly this concept is used when the application consists of several components and these components can either be run on a single machine or distributed over several servers.

The development goal of IBM FileNet P8 components is to support horizontal scaling. The method by which this is achieved varies depending on the component involved, and the method of communication used. As a result, the application delivery and load distribution design can be differ from component to component.

Load distribution

Load distribution, in the context of IBM FileNet P8, is an aspect of horizontal system design allowing load to be evenly distributed to two or more instances of a piece of software. The software instances might reside on a single server or multiple servers or both depending on the needs of the design.

Load balancers are a commonly used method for application delivery and most are suitable for use with IBM FileNet P8. A load balancer is a dedicated piece of hardware or software which inspects incoming traffic and routes it to the appropriate software instance. All of the services in P8 which rely on the HTTP protocol will use either hardware or software load balancer. For reasons beyond the scope of this book, the most common configuration is use both to manage HTTP traffic.

Some applications support load distribution in software only. Often referred to as a request broker design, this method is used when the decision to distribute the load is based on context or state of the traffic. A load balancer, which cannot interpret the content of a request intended for a broker, must not be used for services designed for a broker, and in some cases might not be supported.

Approach

Horizontal applications are scaled using one of two methodologies which describe how traffic moves though the application tiers.

Stovepipe design

The stovepipe design is the simplest way to distribute load but is less reliable than the bowtie design. A client application makes a connection to a distribution component which routes the request to a presentation layer. The presentation layer communicates directly with a predefined service component which in turn communicates with a pre-determined data component.

This methodology is simple to implement and easy to trouble shoot as the communication paths are known, but the design sacrifices reliability. As each component communicates with the next, the risk of one server failing is cumulative because the design does not allow a different component in the same tier to respond instead. The Component Manager of the Process Engine uses a stovepipe methodology, but because the communication path involves only the Process Engine and the Component Manager, the cumulative risk factor is avoided.

Figure 9-1 shows the horizontal stovepipe scaling.

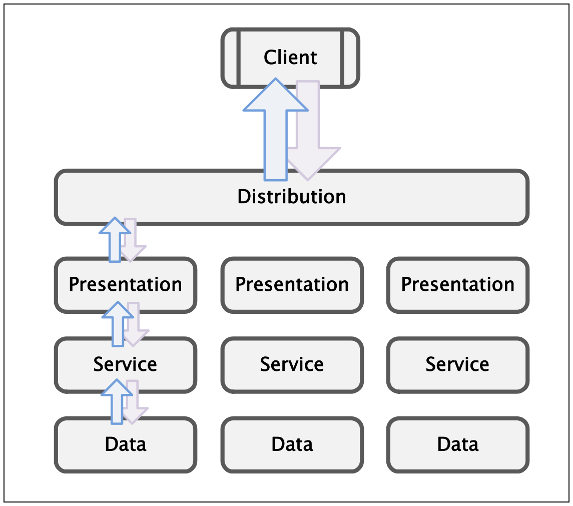


Figure 9-1 Horizontal stovepipe scaling

Bowtie design

The bowtie design improves upon the reliability of the stovepipe design but is more complicated to set up and maintain because it adds distribution steps between every component. The additional distribution steps allow components in each tier to be used in a utility fashion. Thus as long as sufficient capacity is available, a bowtie design can withstand many more failures than a stovepipe design.

While the reliability is high, a sacrifice simplicity is caused by having to configure multiple distribution technologies and requires an understanding of how each technology functions to properly troubleshoot problems. An additional complication is managing application state. If a component fails, the component taking over needs to know where the last component left off. Content Engine and Workplace/Workplace XT are typically scaled using the bowtie method and use persistence and transactions to avoid the component state problem. Figure 9-2 on page 283 shows horizontal bowtie scaling.

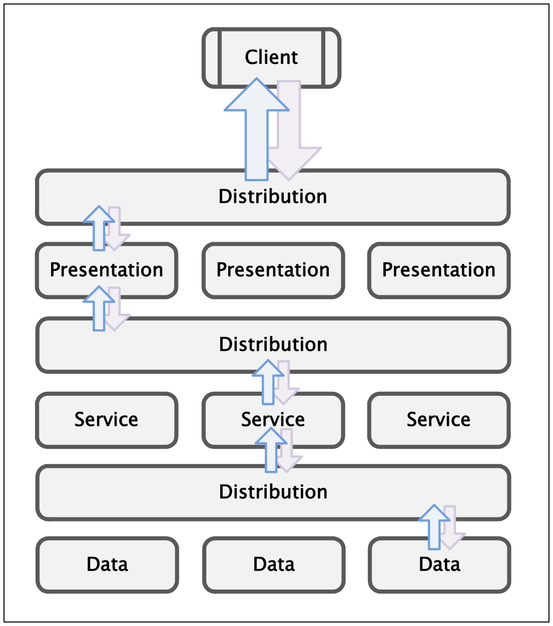


Figure 9-2 Horizontal bowtie scaling

Transactions and persistence

The bowtie scaling methodology requires the solution to maintain state between unlike components and prevent partially completed tasks from harming the integrity of the information. IBM FIleNet P8 uses transactions to prevent cases where a task is only partially completed due to a failure. In some cases P8 implements persistence internally, but because some components are designed to work in conjunction with third party software, such as the application server, persistence is sometimes managed by external mechanisms.

Connection persistence

Connection persistence ensures an established communication path is maintained between the client and the server for the duration of the TCP connection between the client and the load balancer.

Connection persistence maintains the progression of a multi stage communication by knowing the server behind the distribution layer will always be the same until a new connection is established. The connection is defined by the TCP connection from the client to the load balancer. The load balancer impersonates the server by rewriting the network addresses of the packets between the source and destination.

Connection persistence however depends on the communication occurring over a single connection, that the state does not depend on subsequent connections, and that the communication is not arrive over multiple connection streams. Session persistence is more sophisticated method which addresses these limitations. Though it is unnecessary, Process Engine Internet Inter-ORB Protocol (IIOP) traffic will work with connection persistence.

Session persistence

In the context of P8, session persistence is used to ensure a web session started on one client (web browser) maintains state with the server which started the session. Modern web browsers use multiple streams, multiple connection sequences and protocol specific behaviors which make session persistence necessary for P8 user interfaces and web services to function optimally.

Session persistence in an a J2EE application is managed using a JSESSIONID cookie which is created when a browser first connects to a Java servlet or JSP. This key can be used to identify a users session and allows the load balancer to send traffic with the same JSESSIONID to the same server.

9.3.2 Vertical scaling: Scale up

Vertical scaling, or *scaling up*, is achieved by moving an application to a more powerful server. Vertical solutions require fewer machines than horizontal solutions because each server manages more traffic, making the day to day maintenance easier. The drawback to this approach is the server upgrade process when application binaries and data are moved to the larger server. Moreover, if demand outpaces the planned capacity an early upgrade will be required. The risk can be minimized if processor cores and memory can be added to an existing server, as this does not require a re-installation of the application.

An application might or might not take full advantage of the server it runs on and can require tuning the application or the operating system. Most commercial software is designed to require administrative intervention to scale vertically. The application servers supported by P8 require changes to the Java Virtual Machine configuration to scale to the capabilities of the server. Application tuning is explored in greater detail in 9.5, “Tuning the IBM FileNet P8 Platform for performance” on page 313.

9.3.3 Server virtualization

Server virtualization allows multiple instances of an operating system to run concurrently on a single physical server. There are many competing technologies for server virtualization, each with different capabilities and strengths.

Virtualization methodologies usually fall into one of two categories, hardware isolation or software abstraction.

Software abstraction

Software abstraction is the most common form of server virtualization and is implemented in one of two ways. Full virtualization where the hardware is emulated using software and paravirtulization where the virtualized operating system has more direct access to hardware.

As with most computer technologies, software virtualization is implemented in layers. In an unvirtualized server, the operating system kernel runs directly on the hardware and programs communicate with the kernel to access memory, cpu, and peripherals. Software abstraction de couples the operating system from the hardware and provides great flexibility and granularity. The abstraction layer however adds overhead and imposes a performance penalty. The implementation of the virtualized infrastructure heavily influences the performance of the virtualized operating system.

In a fully virtualized server, a second operating system kernel runs in a program called a hypervisor, emulating the hardware. The operating system running in the hypervisor interacts with a virtual machine in the same way it does with a physical machine. The hypervisor translates requests from the virtual machine to the hardware. Fully virtualized operating systems run natively and only interact with the virtual machine and not the hypervisor directly.

In a para-virtualized design the operating system still runs in a virtual machine that has direct access to the hypervisor, and sometimes the hardware. By bypassing the emulation layer, a paravirtualized operating system will usually outperform an equally configured fully virtualized operating system. The drawback to a para-virtualized machine is having to run an operating system designed to support the calls to the hypervisor.

Hardware isolation

Hardware virtualization divides the resources of the server at the hardware level, thus eliminating the need for a hypervisor. Without the overhead of hypervisor, each virtual machine can run a native version of the operating system at full speed. Although hardware isolation has performance benefits over software isolation, it does not support the ability to share a resource, such as a processor core, between virtual machines. Hardware isolation is found only on the higher-end server hardware.

Virtualization is used to address the following challenges:

- Allows applications that are not supported to run on one server to be co-located on the same physical machine by separating them on individual virtual servers.

- Improves the flexibility in provisioning hardware resources to applications. Components that run in a virtualized environment can more easily be moved to a new server.
- Provides a shared platform using fewer physical servers. Use virtualization to partition the servers into logical units into which the individual applications are installed, thus providing data segregation comparable to a scenario where dedicated physical servers are used.
- More effectively utilizes processor and memory resources, for example two physical servers which run at 25% capacity collectively leave 150% of the available processing power unused. If the two workloads when virtualized on a single server use 50-60% of the available processing power leaving only 40%.

Although IBM supports running IBM FileNet P8 in a virtualized environment, organizations pursuing a virtualization strategy are strongly urged to load test virtualized infrastructure intended for production. The liabilities of some virtual infrastructure designs impose significant performance penalties.

9.3.4 Load balancing

P8 is designed to take advantage of several different load-balancing technologies. Load-balancing is a term to describe the distribution of traffic to more than one instance of an application. Load-balancing is necessary when throughput requirements exceed the capability of a single server.

> **Note:** Most load balancers support connection or session persistence. Vendors however have different terms to describe this feature and might refer to the capability as sticky sessions or session affinity.

This section describes various load-balancing methodologies which can be used with IBM FileNet P8.

Hardware load balancer

A hardware load balancer is an appliance specifically designed to distribute network traffic to more than one instance of an application. The term hardware load balancer can also be used to describe load balancers implemented in software but implemented on a closed platform, not intended for general purpose computing.

In a typical configuration, a load balancer will distribute traffic to multiple servers and functions by impersonating a single server. Network traffic directed to a load balancer is inspected and a load-balancing decision is made based upon the content of the traffic in the status of the servers the load balancer is impersonating. A load balancer uses a singular IP address to accept traffic which is sometimes called a virtual server or virtual IP (VIP). The systems the load balancer accepts traffic for are collectively referred to as a resource pool or server pool. Each resource in the resource pool is monitored to ensure the load balancer is not direct traffic to an unavailable service.

All hardware load balancers can inspect traffic at the transport layer or layer 4, also called the transmission control protocol (TCP) layer. The layer 4 rule makes balancing decisions based upon the status of the inbound connection. More sophisticated load balancers support routing decisions based upon application layer code, with the broadest support being available for the HTTP protocol. At this layer, a load balancer can enforce session persistence by inspecting the JSESSIONID cookie in the HTTP header. when the load balancer encounters a JSESSIONID it compares the value against previous routing decisions, and sends the traffic to the resource initially chosen to receive the traffic.

When choosing a hardware load balancer for an IBM FileNet P8 system, it must be guaranteed that it meets at least the following criteria:

- Support for TCP and UDP
- Support multiple virtual IP addresses with different rules
- Support session persistence, (session affinity or sticky sessions)

Load balancers typically sold in pairs and installed in a cluster configuration to ensure the load balancer does not become a single point of failure.

Software load balancer

A *software load balancer* is an application, with similar capabilities to a hardware load balancer, but installable on a general-purpose server. This must not be confused as a web server plug-in or proxy.

Web server plug-in

An application server is often implemented in conjunction with a web server. A specialized plug-in written to support a specific application server works similar to a load balancer. However, plug-ins are restricted to HTTP. The plug-in is configured to be aware of the applications deployed in the application server environment and can make fine grained balancing decisions.

An additional benefit provided by the web server plug-in is the ability to off load functions which do not perform well in Java. The Java language reclaims memory by identifying objects which are no longer in use. This process is called garbage collection. During garbage collection, Java halts all processing and begins reclaiming memory. Every time a client connects a new object which manages the network traffic is created. A web browser will make several socket connections to the application server, transfer some information, and after a period of time, disconnect. because this process repeats itself often it places the burden on the garbage collector. When a web server plug-in is used the burden of managing the socket connections is off loaded to the web server. By reducing the object turnover, the garbage collector runs less often resulting in fewer application pauses.

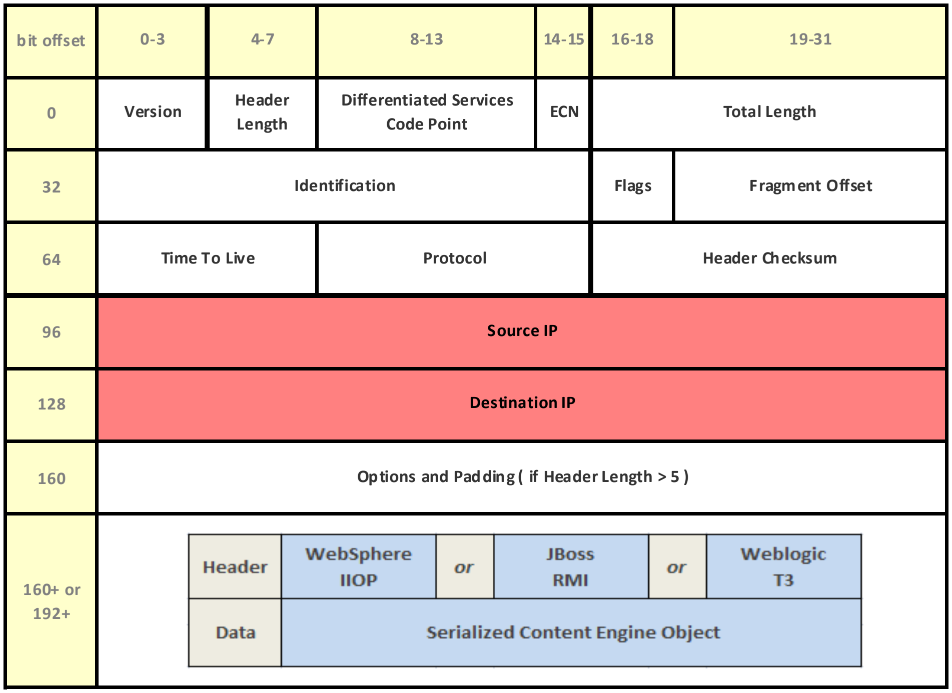
Java object load balancing

Routing EJB requests across a load balancer might cause undesired results and will behave differently depending upon the choice of application server. Figure 9-3 on page 289 illustrates an IP packet containing a Java object. The frame at the bottom represents the data payload containing the object.

A load balancer works by making changes to the packet and forwarding it to an available server based on rules defined by an administrator. Rules are defined by layers which correspond to the Open Systems Interconnection model. A layer 4 operates at the network layer by changing the Source IP and Destination IP of packets as they arrive at the load balancer. The change causes the client to believe the load balancer is the server, and the server believes the load balancer is the client. Some load balancers have the ability to inspect deeper into the packet and make more intelligent balancing decisions.

From a network perspective, the data layer, or payload, contains application-specific information, for example, a packet from a web server will contain information about HTTP in the data layer. Load balancer can inspect improperly interpreted HTTP and can make decisions based on the state of the protocol. However, the payload of an EJB packet is a binary representation of a Java Object. Java has the ability to transmit a binary image an object over a network, but it requires the class file used to generate the object to interpret the data. The design and intent of a load balancer prevents it from making routing decisions based upon information in the EJB payload.

While the initial connection request can be load balanced, only the port used to negotiate the initial context (the lookup of the content engine server) must be configured in this way. The advantage of this configuration is allowing the EJB communication to function as designed, and simplify the configuration of naming lookups, process which occurs before an EJB connection is established.



bit offset	0-3	4-7	8-13	14-15	16-18	19-31
0	Version	Header Length	Differentiated Services Code Point	ECN	Total Length	
32	Identification				Flags	Fragment Offset
64	Time To Live		Protocol		Header Checksum	
96	Source IP					
128	Destination IP					
160	Options and Padding (if Header Length > 5)					
160+ or 192+						

Header	WebSphere IIOP	or	JBoss RMI	or	Weblogic T3
Data	Serialized Content Engine Object				

Figure 9-3 Packet structure

9.4 Scaling the IBM FileNet P8 core engines

This section illustrates the options for scaling the core engines of IBM FileNet P8.

9.4.1 Workplace/Workplace XT

Workplace includes the Workplace web interface and the Component Manager. You can optionally install Workplace XT which is the preferred user interface. Note, Component Manager is installed with Workplace or Workplace XT.

Workplace/Workplace XT includes:

- Workplace/Workplace XT web application interface
- WebDav servlet
- Process Orchestration Web Service servlet (P8BPMWSBroker)
- Process Engine REST Web Service servlet

The scaling of Workplace XT is described in the following section. The scaling of Workplace is not discussed because it is similar to that of Workplace XT.

Application Engine and Workplace naming convention: Application Engine is the official name for Workplace. Application Engine does not equate to Workplace XT. Both Workplace and Workplace XT support a common set of functions but differ in other areas. For consistency of the terminology used in the book, we use Workplace instead of Application Engine throughout the book.

Workplace XT web application

Workplace XT and the WebDav, Process Orchestration Web Service, and Process Engine REST Web Service servlets are deployed as a WAR file to an application server and accessed over HTTP. These service can scale vertically, provided the application server is tuned to do so. These components are most commonly scaled horizontally between two or more servers.

Note: For availability reasons, avoid the stovepipe approach. A stovepipe configuration will cause a single failure to cut the throughput to the entire application and not just the failed component. Additionally the number of servers required must assume one or more of them will offline. In a two server configuration for example, a single system must be sized to provide acceptable throughput when only one server is active.

Workplace XT requires the use of session persistence, also known as sticky sessions. This term refers to a load balancer behavior which causes traffic having a JSESSIONID to always be routed to the same application server. the application server uses the JSESSIONID to match the inbound request to an active HTTP session object. The HTTP session object contains information pertinent to the user which must survive between page loads. This must not be confused connection persistence which will only route based upon the originating IP and the destination port.

Most application servers offer a feature which replicates the HTTP session object to a predefined set of application servers. When a failure occurs, the session object replicated from the failed node is used to maintain state as the user navigates the web application. WorkplaceXT and its associated servlets contain logic which maintains session information without needing to rely on session object replication. As such, this feature does not benefit Workplace XT or the system servlets deployed with it.

Hardware load balancer

Figure 9-4 on page 291 illustrates using a hardware load balancer to distribute the load across a farm of Workplace XT instances.

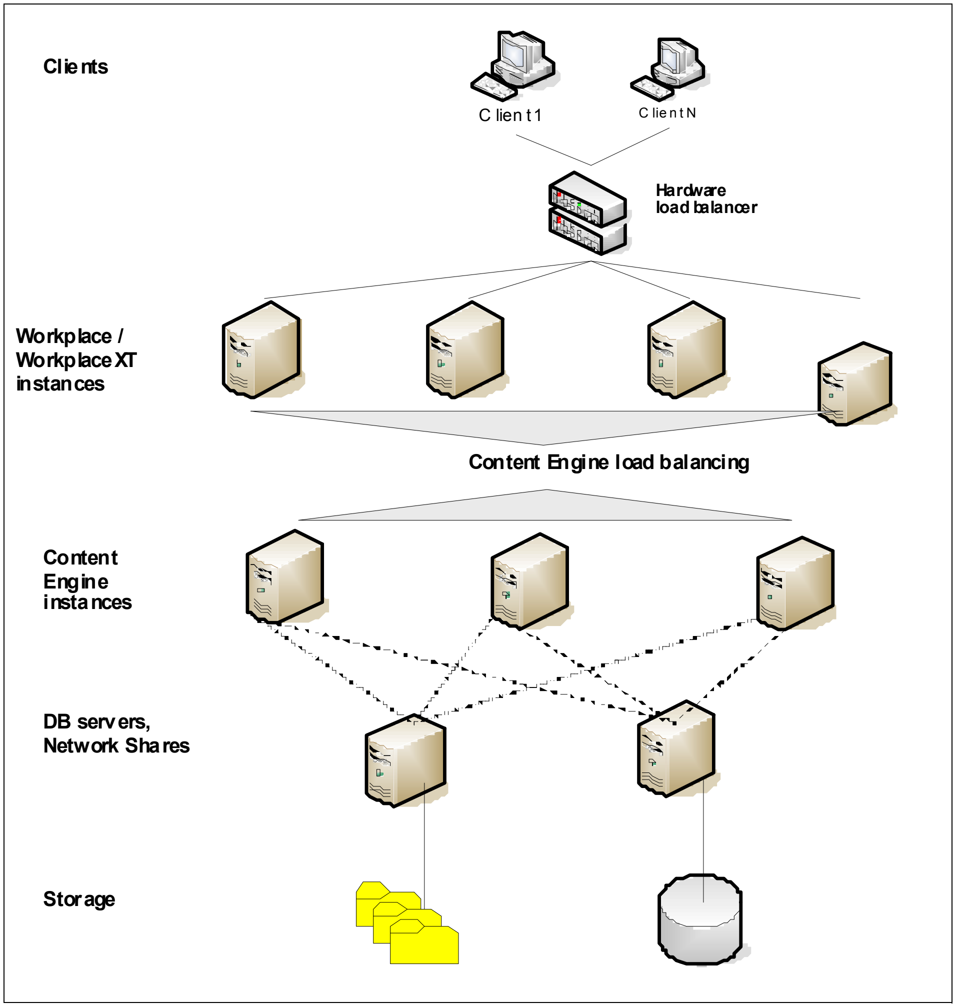

Clients
Client 1
Client N
Hardware load balancer
Workplace / Workplace XT instances
Content Engine load balancing
Content Engine instances
DB servers, Network Shares
Storage

Figure 9-4 Hardware load balancing for Workplace/Workplace XT web application farm

A file system accessible to each instance of Workplace XT is needed for storing configuration data. This is most typically implemented as an NFS or CIFS share. This share must not be provided by any of the Workplace XT servers as this represents a single point of failure. It is advisable to use a highly available NAS device for the purpose. This device need not be a stand-alone device, and the same NAS used to the content provided by the Content Engine can also provide the share.

Web server plug-in

A web server plug-in can also be used to load balance Workplace/Workplace XT. This configuration is useful for incorporating static content, server side includes, or languages other than Java such as Perl, PHP, or Python. Production installations of Workplace/Workplace XT rarely implement load-balancing using a web server plug-in alone, and instead commonly take advantage of both the web server plug-in and a hardware load balancer to simplify the load balancer rules, and shield the application server frequent socket turnover.

Component Manager

The Component Manager runs on the same server as Workplace or Workplace XT, but in a stand-alone Java Virtual Machine. The Component Manager provides the following functionality:

- Dispatches work requests in Component Manager queues to the configured Java classes.
- Allows interaction with configured Java Messaging System (JMS) queues to write JMS messages from within a workflow.
- Performs outgoing web services calls, which are requested by processes that are executed on the Process Engine using system steps with Invoke and Reply instructions.

A single instance of the Component Manager is always bound to one Process Engine-isolated region; therefore, a separate Component Manager instance must be started for each isolated region that the Process Engine uses, in case component integration or process orchestration is used in the processes for that region. Multiple Component Manager instances can be configured using the Process Engine Task Manager.

By default, the Component Manager uses the Content Engine Java API and the WSI transport to connect to the Content Engine and the Process Engine Java API to communicate with the Process Engine. It is supported to configure the Component Manager to use the EJB transport for the Content Engine communication. This is the preferred setup when JMS message queues or if components are used that use client transactions based on the Java Transaction API (JTA) or must get passed the JAAS security context. In this case, we recommend using a separate instance of the Component Manager for each queue using the EJB transport to allow for fine-grained security control.

Using multi threading for components

In the Process Task Manager, you can define how many simultaneous execution threads are used for the component. A thread-safe implementation of the component is required if more than one thread is configured. If thread safety cannot be ensured, it is possible to execute another instance of this component in a separate Component Manager entity.

It is also possible to define multiple Component Manager queues for the same Java Component. However, if these queues are configured to be processed by a single instance of the Component Manager, they are all executed as separate threads in the same Java process by the Component Manager. Thus, defining multiple queues for the same component and executing them in a single instance of the Component Manager does not avoid problems regarding thread safety.

Running multiple instances of Component Manager

Each Java component configured in the Component Manager is executed in a separate Java process spawned by the Component Manager. The Process Task Manager allows the configuration of multiple Component Manager instances on the same server. For each instance, it can be configured which component queues that processes, which is the preferred approach if components are not thread safe and must be executed in separate Java processes.

Scaling web service call requests by processes

If process orchestration is used heavily, running multiple instances of the Component Manager allows you to scale the throughput for outgoing web service calls that active process instances initiate. Each Component Manager instance can be configured to process the outgoing web services queue (WSRequest). If required, different filter patterns can be defined, for example to ensure that high-priority web service requests are exclusively processed by certain Component Manager instances.

Because incoming web service requests for processes are managed by the P8BPMWSBroker, which is part of the web application (Workplace XT), the throughput for processing incoming web service requests can be increased by farming the web application.

In both cases, verify the Process Engine can handle the additional load caused by the increased web service requests using the Scout sizing tool

Workload distribution

In general, the Component Manager instances poll their work from the configured component queues rather than pushing out requests. An exception to this rule applies when Process Engine notifies a Component Manager instance that new work arrived over the Component Manager event port. This feature can be used to configure a large polling interval for the Component Manager and ensures that new work items are nevertheless processed quickly by the Process Engine notifying the Component Manager. In terms of scaling, it is not required to implement a load management for this notification. It is more efficient to use a smaller polling interval if a large amount of work for the Component Manager is expected, which must be the reason why this component is supposed to be scaled; otherwise, Component Manger instances can run in parallel to increase the number of requests that are processed.

The Component Manager configuration includes the link to the Workplace/Workplace XT web application, which communicates with the Process Orchestration web services servlet. Therefore, in a farmed environment, the virtual address of the load balancer or the http proxy must be configured for the Component Manager instances.

9.4.2 Content Engine

The Content Engine is deployed in a J2EE application server as an EAR file. The EAR file contains the Content Engine EJB, the IBM Axis2 Web Service stack for for the Content Engine (CEWS) and for the Process Engine (PEWS). Content Engine supports the same approach for scaling J2EE applications.

Like the Workplace/Workplace XT, horizontal scaling is the preferred option for applications that run in the context of a J2EE application server because it also provides high availability.

Content Engine, being an EAR, has several components, each with a different load balancing methodology. Figure 9-5 on page 295 shows how hardware and EJB load balancing are used to scale the Content Engine.

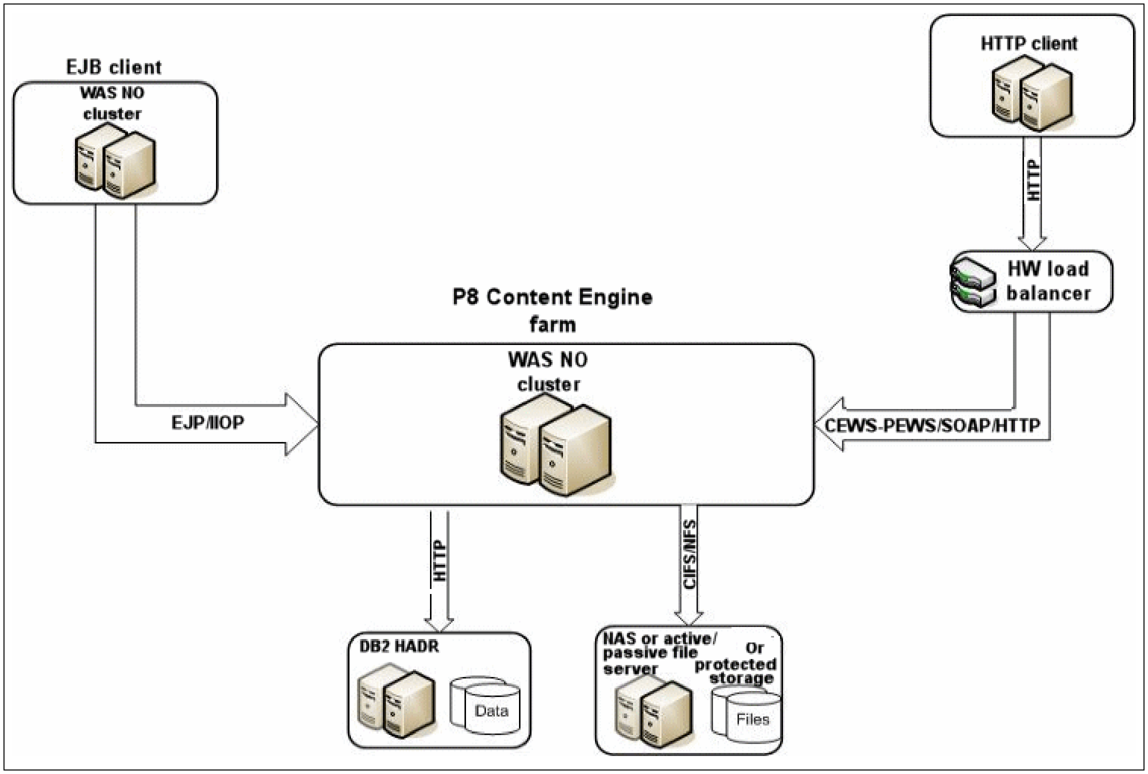


Figure 9-5 Recommended load balancing configuration

Content Engine EJB

The Content Engine EJB is implemented as a set of stateless session beans. A stateless session bean performs work on behalf of a client and relies upon services provided by the application server. A client application using the Content Engine API specifies a URI which indicates to the API how to connect to the Content Engine server. Within the API the client creates an instance of a Content Engine Java object and transmits it to the Content Engine server to be processed. This process requires a TCP connection from the client and server for the duration of the time needed for the object be created, sent to Content Engine, the work performed, and the results returned. Session beans are stateless but only on a per connection basis. This requires that after an object is sent to the server the methods of that object must run, must run on the same server.

The load distribution for EJB traffic is a function of the application server. Thus the use a hardware load balancer is not necessary. Content Engine servers are deployed in groups called clusters. The cluster is responsible for workload distribution. The EJB communication process differs between application server vendors, but they all follow the same basic procedure. A specialized URI is used by the EJB client containing the host names and addresses of the EJB servers. Upon reply, a member of the EJB cluster is elected to handle communication until the network connection from a client terminates.

In the case of IBM WebSphere Application Server, EJB communication occurs over IIOP which is an implementation of General Inter-ORB Protocol (GIOP) over TCP. In cases where Content Engine is scaled horizontally using a cluster, the communication begins when the client causes the Content Engine API to create the initial context (the initial context provides a Context object which is used to perform a JNDI lookup for the Content Engine EJB using the "/FileNet/Engine" context name). The corbaloc URI, runs on a dedicated port and is used to locate a resource which can respond on behalf of the cluster. The sequence completes when the properties needed to locate a Content Engine instance are returned. Several sources can be used for the corbaloc URI, including WebSphere deployment manager, or an available WebSphere node agent. In most cases it is best to use the host and IP addresses of application servers where Content Engine is deployed. After the location of the Content Engine service is established communication occurs over a different port called the ORB_LISTENER_ADDRESS. The ORB_LISTENER_ADDRESS is dynamically assigned by the application server. The client will maintain the TCP connection to this address for as long as the client is alive but there's no guarantee the address will be the same on subsequent connections.

While a load balancer can be used to simplify the URI and load balance the initial context creation process, ultimately the ORB_LISTENER_ADDRESS will be used from that point forward and bypass the load balancer completely.

Note: In cases where a firewall must be installed between the Content Engine and the Workplace/Workplace XT, the ORB_LISTENER_ADDRESS can be fixed to a specific address and port. Additionally the firewall administrator must ensure the firewall is configured to allow connections to be persisted from the Content Engine client to the Content Engine server. Most firewalls are configured to break these connections by default.

Web Services Servlet

Web services servlet provides access to web services API for Process Engine and Content Engine. It is important to recognize the distinction between the Process Engine web service and the Process Engine Orchestration web service which is provided by the Workplace/Workplace XT. The Process Engine and Content Engine web services are implemented over HTTP and can be scaled using multiple instances and the load balancer implemented either in hardware, a web server plug-in, or both. It is highly advisable to protect all web services with the web server plug-in. There is no guarantee a client implementing a web services API call avoids performance impacting behavior, such as frequently connecting and disconnecting.

Content storage

Table 9-3 shows the options for storing content elements with the Content Engine. See 2.2.5, “Content storage” on page 31 for more information.

Table 9-3 Storage options for Content Engine

Storage area	Description
File storage area	Content stored in a folder hierarchy on a shared file system
Database storage area	Content stored as binary object (BLOB) in the object store database
Fixed storage area	Content stored on a supported fixed content device such as FileNet Image Services, IBM N-Series SnapLock, or IBM Information Archive)

A storage subsystem can become a throughput bottleneck affecting ingestion or retrieval rates. This is particularly important for functions outside the context of the application and inside.

Note: Always assign document classes a storage policy and not a storage area. A storage policy allows an administrator to mark storage areas read only for archival purposes. Storage areas assigned directly to a document class can be marked read-only, but subsequent check in and check out of content of the same class will never be routed to a new storage area.

File storage area

A file storage area must be considered as a first choice for storing content if business requirements allow:

- Storage of the content itself represents the largest investment of storage required in a FileNet system. A file storage area offers the broadest choice of storage offerings to keep cost low.
- Storage policies can point to several storage areas and balance the distribution of content to them. Doing so reduces the risk of congesting a storage area’s inbound folder into which each content element is placed prior to being moved into the final location. A file storage area allows an administrator to control directories where content is stored and masks changes made to the underlying storage infrastructure from the FileNet

application. For details about this process, refer to *IBM FileNet Content Manager Implementation Best Practices and Recommendations*, SG24-7547.

- File storage areas can be implemented on NAS storage, such as IBM N-Series to take advantage of advanced features, such as snapshots, replication, and data cloning.

> **Note:** Some file systems are designed to perform well for small files where others are designed to perform well for large files. ECM applications store many small files which are accessed randomly. File systems not designed for small file storage will demonstrate performance problems as the number of files increase.

Database storage area

Using a database storage area requires that all content is transferred into a *single* object store database. Whereas it is possible to scale the uploading units by adding Content Engine servers, all content elements must be received and stored by the database engine. This traffic uses the same infrastructure components of the J2EE server, such as the metadata, for example the JDBC connection pool to the database. Compared to the file store example previously discussed, there are fewer options to establish parallel communication channels. Thus scaling relies heavily on the capability of the underlying database system.

A database storage area delivers the benefit that content and metadata is stored in a single database, which makes backup and restore scenarios easier because there is no need to ensure synchronization between a file system or fixed content device and the metadata database. Organizations that choose to use the database as the storage area must configure the database so that the content is stored separately from the metadata.

Fixed storage area

Fixed storage areas are often used if the system must meet data retention requirements. In many cases, it is easier to ensure, on the level of the storage device, that content that is written one time cannot be changed later on for a defined period of time (referred to as the *retention period*). Based on the concept of the content life cycle (refer to 7.2.2, “Document life cycle” on page 173), the Content Engine allows the content to be moved to such a device, for example when a status is reached where content is archived for a certain retention period.

Fixed storage areas, require a staging area accessible to every instance of Content Engine. New content elements are placed into this staging area before they are moved to their final destination on the fixed content device. For high ingestion rates, multiple staging areas can be load balanced using a storage policy, although this does not change the fact that a single fixed-content device finally stores all of the content elements.

Benchmark

In various benchmarks, the Content Engine delivers excellent performance and scales extraordinarily well. Content Engine demonstrates near-linear growth in throughput when additional instances were added to the Content Engine farm. Refer to the white paper *IBM FileNet P8 4.0: Content Engine Performance and Scalability*[1] for details.

A cluster of 16 Content Engine servers was benchmarked using a single object store. Results indicate that the performance of the database becomes a bottleneck before Content Engine demonstrates nonlinear scalability. Database technologies such as IBM DB2 pureScale, and using multiple object stores on different databases, can be used to scale Content Engine even further.

9.4.3 Full-text indexing

The P8 Platform provides the ability for full-text indexing on documents and their metadata through Content Search Services and the Legacy Content Search Engine.

Two major functions of the Content Search Engine are:

- Create full-text index information
- Execute content based retrieval (CBR) queries utilizing the fulltext index

Content Search Services

This section discusses Content Search Services (CSS)

Indexing scalability

The Content Engine allows for multiple CSS servers to be configured for a P8 site. A CSS server might be configured in index, search, or index and search mode. Only CSS servers configured in index or index and search mode can perform full text indexing. CSS natively supports farming of index servers to allow for greater throughput of indexing requests. To enable farming, all index servers in the site must have access to the index areas. An index area consists of index files that contain index information for the objects that belong to the same indexable base class or subclass of the base class. An index area begins life with one index file and create indexes until the index file's capacity is reached and will consequently create another index file for additional indexes. An index file can only be accessed by only one index server at a time. The performance benefit of

[1] Available on request. Contact your IBM representative for information.

multiple index servers will be seen when there are multiple index files. The way to ensure multiples index files is to either use multiple index areas or to implement index partitioning.

Index partitioning is the grouping of index information into separate indexes based on object property values. CSS allows for index partitioning using a maximum of two properties, one string property and one date property. If an object contains is partitioned using a string property, all objects that have the same value for the property will be stored in the same index. For example, objects containing the string property *Department* with the value *Legal* will be grouped together in as separate index. This will result in multiple index files for a given index area based on the different values of the *Department* property. On creation of the index, an index server is assigned to the index given a lease time for exclusive write access to the index. The index server will relinquish those rights only if the following conditions occur:

1. The index server becomes unavailable and remains unavailable past the expiration of the lease time.
2. The index server's lease time has expired and the index server has been deemed unfit to maintain ownership based on the total number of indexes owned compared to other index servers in the farm.

The second condition helps with balancing the load across all the registered index servers in the farm. In either instance, a new index server is assigned and given a new lease time and exclusive write access to the index. Potentially, every index file in the index area can be maintained by a separate index server.

Search scalability

Only Content Search Services (CSS) servers configured in search or index and search mode can be used to search indexes. Multiple CSS servers can be registered with a P8 domain to form a farm of search servers. When a Content Based Retrieval (CBR) query is submitted, the Content Engine selects the dual-mode (index and search) server that most recently updated the index as the default search server. If the server is unavailable or is not configured in dual-mode, a random search server is selected and the query is executed with the selected server. As with indexing, all search servers need to have access to the index areas in their P8 site to perform searches.

Legacy Content Search Engine

This section discusses the Legacy Content Search Engine.

Indexing scalability

Autonomy K2 allows you to configure multiple K2 servers to work as a cluster to provide high availability and scaling. This configuration improves the throughput for indexing requests at the level of Autonomy K2 because multiple indexer processes can work in parallel on the different nodes of the Autonomy K2 cluster. Figure 9-6 illustrates the Scaling out Content Search Engine indexing.

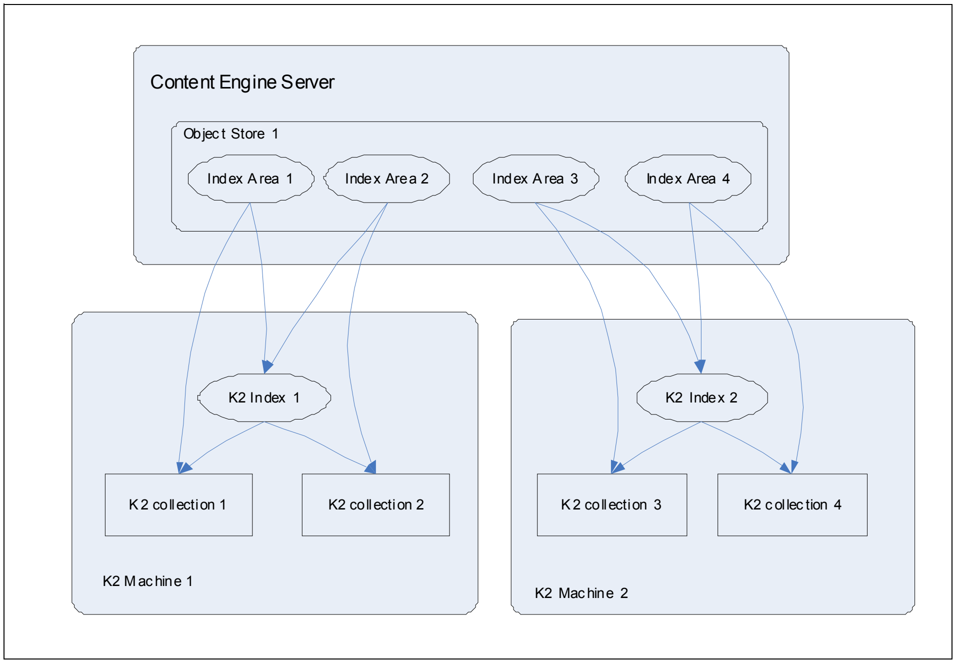


Figure 9-6 Scaling out Content Search Engine indexing

Each server in the Autonomy K2 cluster is configured to handle one or more *Index Areas*. Using at least two Index Areas on each Autonomy K2 server improves the throughput due to an increase in concurrent writes. Figure 9-6 illustrates a configuration with multiple Autonomy K2 servers and index areas. Each search server handles two search areas in this example.

The Content Engine CBR Executor hands over an indexing request to any of the servers that are configured for the appropriate Index Area in a random fashion, thus distributing the load.

Note: Currently, only one CBR dispatcher process can be active for an IBM FileNet Content Manager site. Content Manager provides a feature such that for each site, only one Content Engine server runs the CBR dispatcher process. If one is down, it automatically launches a dispatcher from another server to perform the task.

Search scalability

To raise the number of search requests that can be handled simultaneously, the number of Autonomy K2 search servers must be increased. If multiple Autonomy K2 search servers are available, the Autonomy K2 broker process issues the incoming requests to all search servers simultaneously where they get processed concurrently. The Autonomy K2 broker gathers the results that the search servers deliver and returns them to the Content Engine.

Figure 9-7 illustrates the use of multiple Autonomy K2 search servers to scale out content-based retrievals. There is only one Autonomy K2 broker process required (on Autonomy server 1) that accepts the search requests that the Content Engine executes and dispatches them to the search servers.

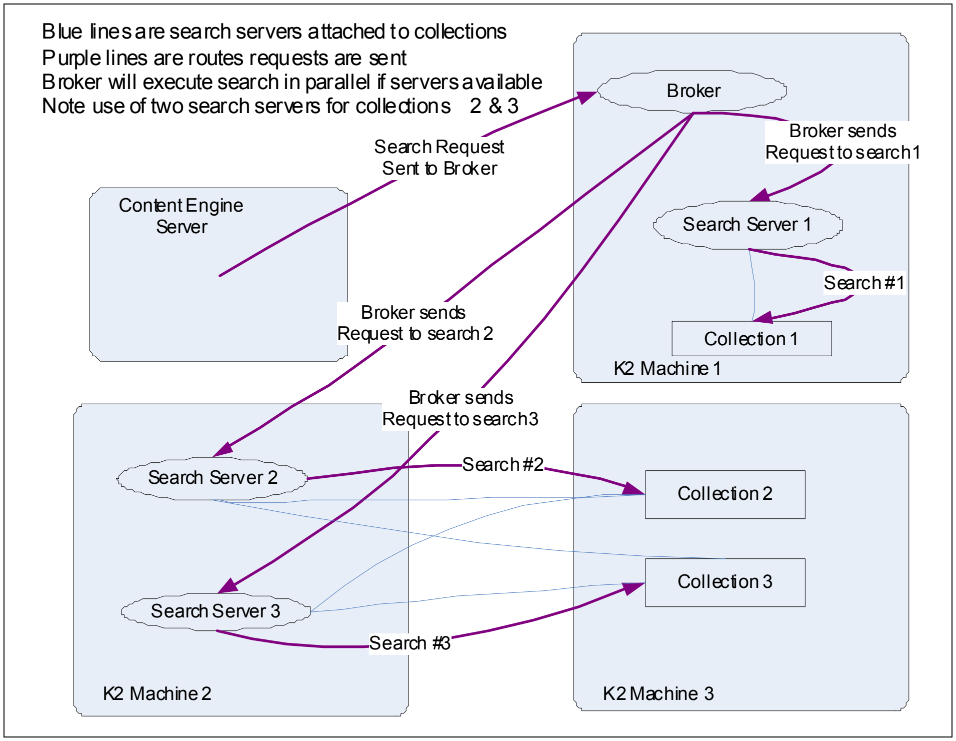

Blue lines are search servers attached to collections
Purple lines are routes requests are sent
Broker will execute search in parallel if servers available
Note use of two search servers for collections 2 & 3
Broker
Broker sends Request to search 1
Search Request Sent to Broker
Content Engine Server
Search Server 1
Search #1
Collection 1
K2 Machine 1
Broker sends Request to search 2
Broker sends Request to search 3
Search Server 2
Search #2
Collection 2
Collection 3
Search Server 3
Search #3
K2 Machine 2
K2 Machine 3

Figure 9-7 Scaling out Content Search Engine retrievals

We recommend using advanced technologies, such as the classification module, to extract useful metadata from document fulltext instead of purely relying on a fulltext search engine. The availability of such metadata is key in an Enterprise Content Management strategy because it is the fastest way to make content accessible for a wider audience of users. Being able to use metadata, sometimes in conjunction with fulltext information, is the superior concept of relying on the fulltext information as the primary source for finding content.

9.4.4 Process Engine

The Process Engine is a Java application, but it does not run in the context of a J2EE application server. Nevertheless it supports both horizontal and vertical scaling to respond to increasing system demands. The recommended configuration for Process Engine is horizontal scaling. This section highlights the options available for scaling Process Engine:

- Single Process Engine server
- Farming Process Engine servers
- Independent Process Engine servers

Single Process Engine server

If the Process Engine system consists only of a single server, scaling vertically is the only available option. The Process Engine functions are executed by worker threads that are controlled by a central broker. This broker manages worker threads that dispatch the process instances (*work objects*) and the background threads, which manage the email notification and similar tasks.

If additional CPU cores and memory are available, it is possible to increase the number of parallel threads, which enables the Process Engine server to handle an increasing number of work objects.

Farming Process Engine servers

The Process Engine supports farming of Process Engine servers. In such a configuration, multiple Process Engine servers access a shared database that stores the information, such as work objects and queues.

Figure 9-8 on page 304 shows a configuration where two separate applications access a shared Process Engine farm, which is a common configuration because both applications can use different isolated regions, which ensures that work objects do not interfere.

The farming capability introduced IBM FileNet P8 4.0 allows organizations to implement the same concept of scaling horizontally across all core engines of the Platform. In addition, it provides high availability without the need for spare servers running idle.

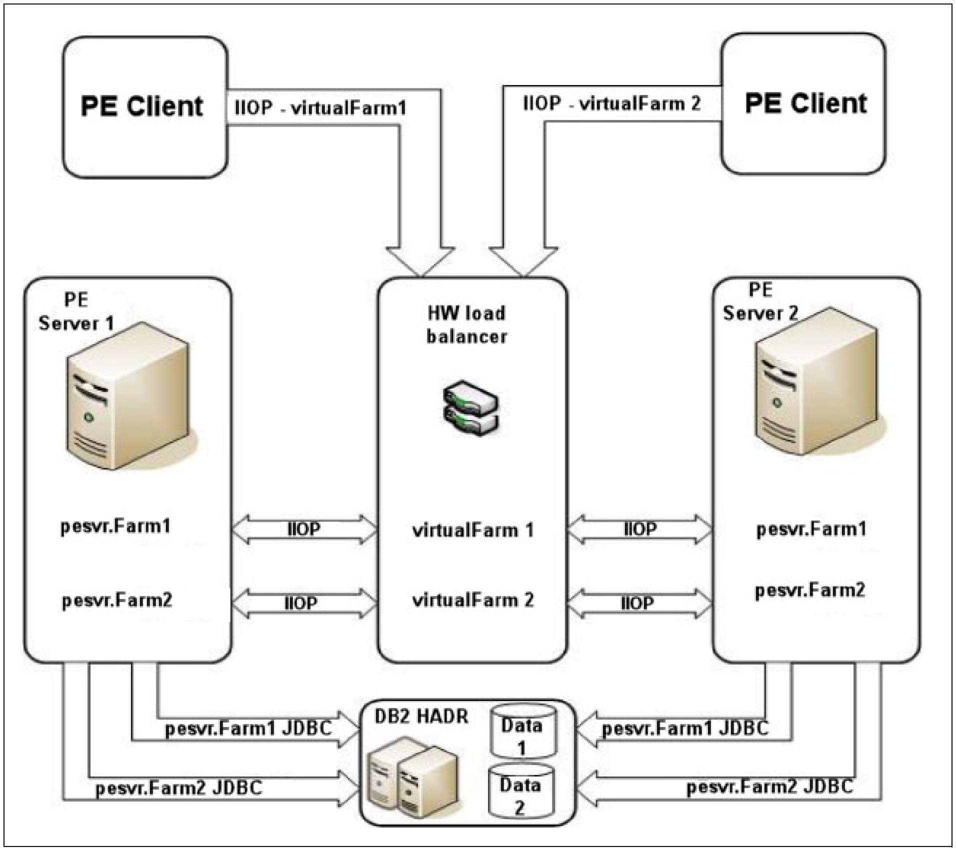


Figure 9-8 Two node Process Engine server farm with multi-tenancy

Farming the Process Engine servers requires a load balancer to distribute the incoming requests. Because the connections are stateless, connection affinity is not required. Details about the installation and configuration are in the IBM FileNet P8 High Availability Technical Notice.

Using the Process Engine Task Manager, all servers in the Process Engine system can be managed from a central location. This central management includes starting and stopping the Process Engine software on single nodes and removing nodes from or adding nodes to a farm.

Farming is also the preferred way to scale the Process Engine. This concept delivers high availability because the remaining servers continue to deliver the Process Engine functionality if a single node in the farm fails.

IBM conducted benchmark tests that proved a near-linear growth on transactions handled by a Process Engine Farm when additional servers were added to the farm. Because the Process Engine scales extremely well, the database that is the Process Engine farm uses might become the bottleneck when scaling horizontally.

Independent Process Engine servers

The Process Engine system can also be scaled by using independent servers or independent instances. As Figure 9-8 on page 304 illustrates, each Process Engine server uses its own database, which can be hosted on an individual database server. The same configuration is possible for Process Engine instances using independent Process Stores (databases).

Scaling with independent servers is an option to scale when the overall architecture is geographically distributed and work is mainly done locally, for example, the server for application 1, the left Process Engine server, and the left database server can be in Los Angeles; whereas, the servers on the right are located in New York to support another application that is used there. The fact that users in each location always work with local servers and a local database delivers optimal performance.

Note: It is recommended that this methodology is used in conjunction with load-balancing. Having singular independent instances or servers without redundancy will mean having to maintain a system with multiple single points of failure.

Two alternatives exist regarding the Content Engine for storing the process-related content objects. Both applications can use a shared Content Engine server farm, or both applications can use separate Content Engine servers. In the example of the distributed environment, separate Content Engine servers are configured, one for each location to ensure that the application can locally retrieve the content.

9.4.5 Summary

The IBM FileNet P8 architecture supports horizontal and vertical scaling for all core platform components to respond to increasing system demand. Benchmarks show that IBM FileNet P8 shows nearly-linear scaling over a wide range for both the Content and Process Engine.

Using the approach of farming, the layers for the Workplace/Workplace XT, Content Engine, and Process Engine, IBM FileNet P8 provides a solution that makes it easy to add resources to a given system. The only thing that you must do is provision additional servers (or additional instances on existing servers, where applicable) with the corresponding platform server component, and reconfigure the workload management component to actively use this component.

9.4.6 Scaling add-on products

There are many products that are IBM FileNet P8 based. In this section, we discuss scaling of some of these add-on (expansion) products and modules:

- Business Process Framework
- FileNet eForms
- IBM FileNet Capture
- IBM Content Collector
- Connectors for Quickr, SharePoint, and SAP
- Case Analyzer
- Process Simulator
- IBM FileNet Image Services

For an introduction to these add-on products, refer to Chapter 2, “Core component architecture” on page 19.

Business Process Framework

The Business Process Framework (BPF) can scale in the same way that the Workplace scales because it has a similar structure. The core components of BPF are:

- The web application that is deployed on a J2EE server
- A Component Manager queue for BPF-related operations that can be triggered by a process executing on the Process Engine

The scaling options for the Workplace found in section in 9.4.1, “Workplace/Workplace XT” on page 289 also apply for BPF. Similarly, we recommend farming the web application to gain benefit from the high availability, which is automatically introduced by this architecture. For the Component Manager that hosts the BPF operations, the best practice is to configure additional instances to handle an increasing number of requests. However, because the BPF operations components are implemented thread safe, it is also possible to configure multiple threads for a single BPF operations queue.

It is important to remember that one BPF application, just like Workplace, can only be configured to use one isolated region. This is no restriction in flexibility, and for the sake of data segregation work objects cannot cross the border between isolated regions. Therefore, it makes sense to implement individual BPF applications for each isolated region.

FileNet eForms

Electronic Forms (eForms) for IBM FileNet P8 is an extension to an Workplace/Workplace XT installation. As such, eForms can be scaled in the same way as a client web application is scaled. Scaling the Workplace/Workplace XT automatically takes care of eForms.

eForms provide several options to integrate with external systems, for example: databases by taking advantage of JDBC lookups or arbitrary systems by using HTTP calls to lookup data. You must ensure that the system that eForms integrates with and the intermediate piece that facilitates this communication (for example, a servlet performing a lookup against a host system) are designed accordingly so that they can handle the increased load that originates from scaling the eForms and the Workplace/Workplace XT.

IBM FileNet Capture

IBM FileNet Capture enables capturing of paper-based documents and faxes and storing them as electronic images in the IBM FileNet Content Manager repository. Capture offers various options to provide metadata for the content, such as automatic extraction, using the ADR module or by manual keying using the indexing application.

Capture was designed to run as a distributed application and supports horizontal and vertical scaling for the different building blocks. Capturing images, performing quality checks, combining images into documents, and adding metadata to the documents before committing them to the repositories can include various steps. The sequence of this capturing process can be expressed as a *Capture Path,* which describes how the images flow through the different processing steps until they are committed as a part of the document. Capture does not require that a Capture Path is set up, but you can work with the different modules in an ad-hoc mode. To ensure that all steps are executed in the desired sequence, we recommend that you define a Capture Path for this purpose.

The approach to scaling Capture to process a larger amount of documents differs for the components that are used. Modules that require manual interaction by a person, such as the scan operation itself, the quality review, indexing, or indexing review steps, can be scaled horizontally by adding additional PC workstations to the installation. Images are scanned in parallel by the scan stations into separate batches that are managed in a database, and the Capture Path ensures that indexing and review stations pull images from these batches for further processing. By adding PC workstations, more people can work in parallel on the different batches.

Modules that perform automated tasks, especially ADR recognition components that extract data from the images using statistical methods can be scaled vertically, and adding processing power to an ADR server improves its performance and increases the throughput. Scaling horizontally by increasing the number of ADR servers is the other option that is available for these parts of the ADR installation.

IBM Content Collector

IBM Content Collector consists of three major building blocks:

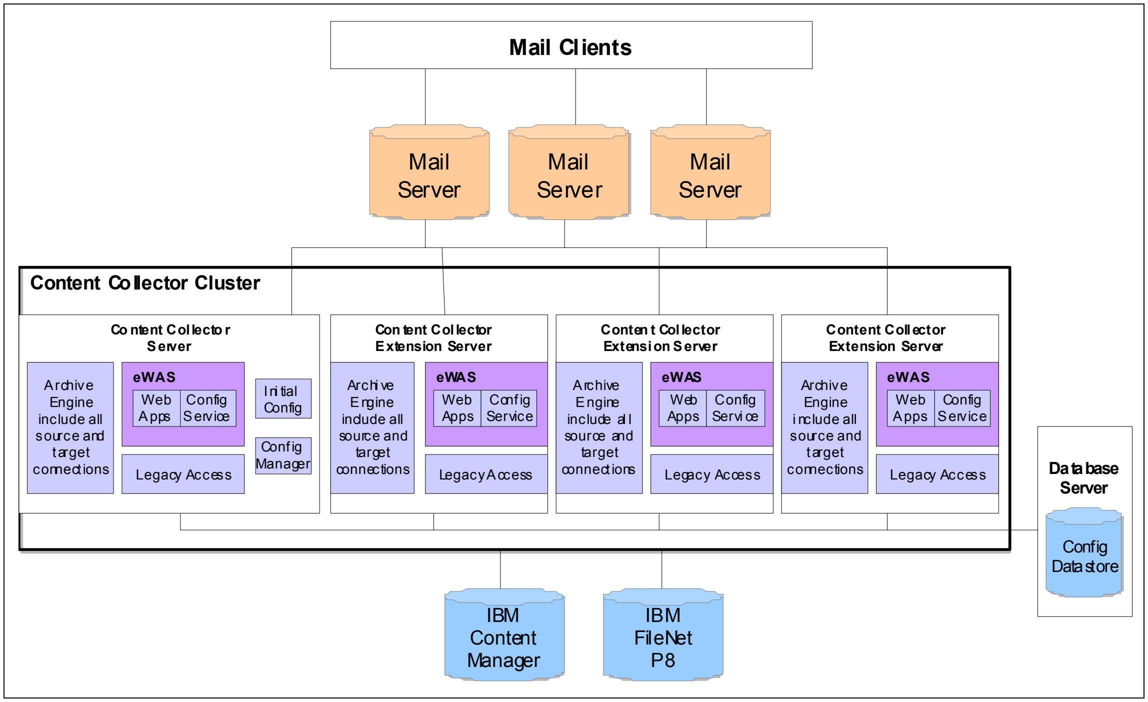
- An archiving engine
- Web applications and configuration service running on an embedded IBM WebSphere Application Server
- Legacy access components

Content Collector supports horizontal scaling for all components to increase the throughput for archiving new content and for serving a larger amount of users.

A horizontal form of Content Collector servers is called a *Content Collector cluster*. Because Content Collector runs on Windows operating systems only, there is no option to scale Content Collector vertically, except for using virtualization. Content Collector is a component that heavily uses the network to connect to the Content Engine for storing content. Therefore, we do not recommend using virtualization to run multiple instances of Content Collector on a single physical server.

Figure 9-9 on page 309 illustrates the setup of an Content Collector cluster. The (primary) Content Collector server runs the core components. In addition, the configuration manager and the initial configuration template are hosted on this system. When additional servers are installed to form the cluster, they use the installation procedure for the expansion server, which only runs the core components.

All configuration for the Content Collector system is stored in a central configuration database that all servers in the Content Collector cluster access.



Mail Clients
Mail Server
Mail Server
Mail Server
Content Collector Cluster
Content Collector Server
Archive Engine include all source and target connections
eWAS
Web Apps
Config Service
Legacy Access
Initial Config
Config Manager
Content Collector Extension Server
Archive Engine include all source and target connections
eWAS
Web Apps
Config Service
Legacy Access
Content Collector Extension Server
Archive Engine include all source and target connections
eWAS
Web Apps
Config Service
Legacy Access
Content Collector Extension Server
Archive Engine include all source and target connections
eWAS
Web Apps
Config Service
Legacy Access
Database Server
Config Data store
IBM Content Manager
IBM FileNet P8

Figure 9-9 Horizontal scaling for Content Collector

Scaling Content Collector: Content Collector performs retrieval by communicating directly with Content Engine; therefore, Content Collector must be scaled as utilization grows. This approach is different from its predecessor IBM FileNet E-mail Manager, which uses the Workplace/Workplace XT to retrieve and display content and thus relies on scaling the Workplace/Workplace XT to ensure speedy retrieval.

Connectors for Quickr, SharePoint, and SAP

The connectors for Quickr, SharePoint, and SAP consist of a repository connector component (the connector for SharePoint document libraries) that transfers content from the collaboration environment into the ECM system and extensions for the collaboration web portal application, which allow direct access objects in the Content and Process Engine from the collaboration environment. These extensions are only used for retrieval purposes from the collaborative web portal.

Collecting and transferring content

The connector for document libraries collects the content from the document libraries in the collaboration environment, transfers it into the Content Engine, and (depending on the configuration) performs other tasks, such as creating a stub. The connector for Quickr document libraries currently allows you to collect content manually, not based on business rules, such as the SharePoint connector. For that reason, scaling mainly refers to the SharePoint connector. This component can be scaled vertically on a single server because the number of threads can be adjusted to the processing power of the underlying server hardware.

Additionally, if multiple Quickr or SharePoint instances are connected to the ECM repository, it is possible to distribute the overall load by installing and configuring individual instances of the connectors for a dedicated set of Quickr or SharePoint instances.

Retrieving content

The connector for SharePoint Web parts or Quickr Web portal forms the component that is used for retrieval purposes. From a technical point-of-view, it uses the Content Engine and Process Engine APIs to connect to the back end. Content retrievals are executed by calling the appropriate functions of the Workplace UI Service.

The retrieval part can be scaled horizontally using a farm of SharePoint servers, for example. The web parts are then installed on each server of the SharePoint farm, and therefore an increasing number of users and requests can be handled. Because the retrievals are triggered by persons working with the collaborative environment, increasing retrievals from the ECM system is typically seen if more users are working with the collaborative environment, so that a farmed environment might already be in place.

Because the content retrieval uses the Workplace/Workplace XT functionality, you must consider the impact of increasing retrievals on the Workplace/Workplace XT load. As illustrated earlier, the Workplace/Workplace XT can easily be scaled horizontally.

Case Analyzer

The Case Analyzer extracts data from the Process Engine event log tables and feeds them into its internal data warehouse. On a scheduled time interval, the data is aggregated into an internal data mart and from this representation OLAP cubes are derived, which can then be analyzed further by OLAP-aware tools, such as Cognos or Microsoft Excel. Converting data from the warehouse into the data mart and calculating the OLAP cubes is a labor-intensive job.

With IBM FileNet P8 release 5.0, the Case Analyzer can be scaled horizontally to deliver higher throughput and faster calculation times. Case Analyzer is a J2EE application which uses IBM Mashup Center and is deployable to a supported application server. Case Analyzer is scaled using the same methodology as IBM Mashup Center.

Process Simulator

The Process Simulator uses a process definition, arrival and work shift patterns, and probabilities for conditional routes to predict the flow of process instances, thus enabling it to perform what-if assessments to eliminate bottlenecks in existing processes or avoid them for planned processes.

The Process Simulator can be scaled vertically by adding additional CPU and RAM to the server on which this component is installed. Because the process simulator does not have to process items on the magnitude level of a production system, it is most likely that this component will not become a bottleneck.

9.4.7 IBM FileNet Image Services

IBM FileNet Image Services is an image management repository that was designed to efficiently handle large amounts of electronic images. The Image Services architecture is optimized to deliver world-class storage and retrieval performance while managing billions of image documents. Content Engine uses Content Federation Services (CFS) to utilize Image Services as a repository for IBM FileNet P8. Figure 9-10 illustrates the basic layers of the Image Services architecture.

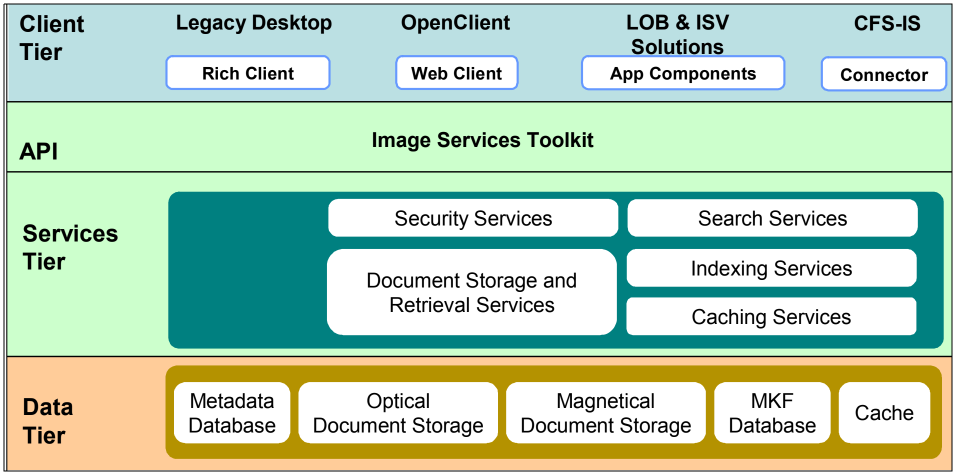


Figure 9-10 Image Services architecture

In Figure 9-10 on page 311, the lower layer is the data tier that Image Services use to store the images that are managed and the data that is related to them. Some important building blocks are:

- The relational database that holds the metadata for the images.
- The multi key file (MKF) database that stores the location for each image on the storage media.
- The magnetic cache regions that provide efficient batch document ingestion and speed up retrieval times, especially for documents that are stored on optical media.
- The optical and magnetic storage and retrieval systems.

Image Services supports a large variety of storage subsystem that can be used to store the images, such as Jukeboxes (optical storage), and several magnetic storage systems, such as disk, IBM DR-550 compliance storage, IBM N-Series with SnapLock, and EMC Centera. For a complete list of supported systems, refer to the *Image Services Hardware and Software Guide:*

ftp://ftp.software.ibm.com/software/data/cm/filenet/docs/isdoc/IS_HW_SW_guide.pdf

In Figure 9-10 on page 311, the services layer provides the services that are required to manage the stored documents. It consists of:

- Document storage and retrieval services that write and retrieve the objects to the storage system
- Indexing services that manage the associated metadata
- Cache services that manage the cache regions that are used
- Security services that provide access control to documents
- Search services that allow you to quickly locate documents based on their metadata

Image Services are designed to work as a distributed application and are implemented as a set of logical services, which means that many of the components that are listed in Figure 9-10 on page 311 can be spread over different servers. In earlier times, when only optical media was used as long term storage, multiple OSAR servers where frequently used because they provided the required number of SCSI ports to connect a larger number of optical jukebox libraries. It is also possible to off load components, such as the indexing services or cache services, to one or more dedicated servers. Therefore, Image Services supports horizontal scaling.

Image Services is also optimized to effectively use the resources that the host server provides. As such, all Image Services component can be executed on a single system that provides vertical scaling. However, for systems that must connect to a large number of optical jukeboxes, the required number of SCSI controllers can be a limiting factor for pure vertical scaling.

9.5 Tuning the IBM FileNet P8 Platform for performance

For the IBM FileNet P8 infrastructure components that run in the context of a J2EE application server, at first glance, there is only a limited difference between the approach of scaling vertically versus horizontally because in both cases a farm of application server nodes is created. However, both alternatives show significant differences in that a hardware resource, such as memory or an I/O subsystem, must be shared between the different entities in the case of vertical scaling (comparable to virtualization).

There can be a benefit for vertical scaling because any communication between the components that run on the same physical machine do not have to travel over the wire. However, this does not apply if all Content Engine instances run on server1 and all Workplace/Workplace XT instances run on a different server2 because there will be no direct communication between the Content Engine nodes but instead the majority of the traffic is seen either between Content Engine and the database and the storage subsystem and between the Content Engine and the Workplace/Workplace XT. Therefore, vertical scaling can show performance benefits if the IBM FileNet P8 core server components Workplace/Workplace XT, Content Engine, and Process Engine are running on the same physical machine, thus providing minimum latency and maximum bandwidth for the communication path. For installations that make no or only limited use of the Process Engine, this concept might be applied to the Application and Content Engine only.

Refer to the *IBM FileNet P8 4.0 Performance Tuning Guide* for additional information and recommendations:

ftp://ftp.software.ibm.com/software/data/cm/filenet/docs/p8doc/40x/p8_400_performance_tuning.pdf

9.5.1 J2EE Application Server

J2EE Application Server performance can be optimized in Java Virtual Machine and through connection pooling.

Java Virtual Machine

The Content Engine and the web application of the Workplace/Workplace XT are deployed into separate instances of a J2EE application server.

One critical aspect regarding performance is the garbage collection (GC). Garbage collection refers to an activity where the Java Virtual Machine (JVM) re-organizes its memory heap, deletes all references to objects that are no longer used, and frees this memory for new objects. The larger the amount of memory that was configured for the JVM, the longer this garbage collection cycle takes and the more server resources are used for the garbage collection. The important point is that during the full garbage collection cycle of the JVM, all applications that run in the JVM do not respond. With 2 GB of memory, the garbage collection cycle can take 30 seconds or longer, depending on the resources that are available for the JVM.

On the other hand, the JVM needs a certain amount of memory to run the application. If there is a small amount of memory available, the JVM either often undergoes garbage collection cycles or throws an out of memory exception and eventually crash the application.

In general, the memory requirements of the Content Engine, and especially the Workplace/Workplace XT, strongly depend on the usage pattern. We recommend starting with a configuration of 1 GB of memory for the JVM of each Workplace/Workplace XT and web application and 2 GB of memory for the JVM of each Content Engine.

To speed up the GC process, we recommend that you configure the generational concurrent GC strategy to reduce to the minimum the pauses that the GC cycles cause. On the downside, this strategy reduces the memory throughput to accomplish the improved GC cycles.

Note: For high-ingestion scenarios, which create a large amount of short-living objects in the Content Engine, it can be beneficial to provide additional space in the tenured generations of the JVM heap by adjusting the ratios accordingly.

It is important to understand that different load patterns might require different JVM tuning to achieve optimal performance, for instance, it can be helpful to configure dedicated Content Engine instances for import purposes that run on J2EE application server instances that are optimized for high-ingestion performance; whereas, other Content Engine instances that are primarily used for retrieval use a different JVM configuration. If high volume ingestion is primarily performed over night, it can be beneficial to use scripts to shutdown the J2EE application server, change the JVM configuration accordingly, and then start the import jobs and switch back to an alternative JVM configuration for the daytime operations.

We highly recommend that you validate and monitor the effectiveness of the JVM performance tuning by using the appropriate tools, such as Tivoli Performance Monitoring (TPM) or similar.

Connection pools

The Content Engine uses the configured connection pools to communicate with the GCD and the object store databases. It is mandatory to adjust the `maximum connections` parameter for the data sources to the expected number of client connections. Refer to the IBM FileNet P8 Performance Tuning guide for detailed formulas to calculate the connection pool size dependant of the client connections expected.

9.5.2 Database

Content Engine and Process Engine use databases to store and retrieve information about content and process objects. It is important to configure the database accordingly to achieve optimum performance.

Database indexes

As a general guideline, it is important to know which queries are performed against the databases to create the required database indexes for preventing full table scans. Both Content Engine and Process Engine support the creation of database indexes through the administrative tools Enterprise Manager and the Process Configuration Console. In many situations, simple database indexes are not sufficient because the queries that the API creates only benefits from complex indexes, such as a combined index. IBM FileNet P8 supports the creation of combined indexes for the Content Engine using the tools provided by the database vendor and for the Process Engine by using the Process Configuration Console.

Index skew

The distribution of values in an index might become uneven, for instance if half of the objects have the same value for an indexed metadata property. This situation is described as *index skew* and results in the database not using the index anymore and performing a full table scan instead, even for searches that actually benefit from the index. By changing the query optimizer statistics strategy, as described in detail in the Performance Tuning Guide, the database can be instructed to use the index for those queries that refer to values in the index that are used only for a few objects.

Statistics collection

It is important to ensure that the statistics collection for the database query optimizer is run periodically and that the tools supplied by the database vendor are utilized, which helps to identify long running queries and suggest additional indexes to remedy these situations. In case the query optimizer is tuned manually, it is important to update the job profiles for the statistics collection on the database accordingly so that the changes are reflected and not overwritten with a default statistics job.

In high ingestion scenarios, using multiple object stores is beneficial because it allows you, at least, to address different tablespaces (DB2 or Oracle) or different databases (MSSQL) that can be physically located on different disks on the database server. Object store databases can also reside on different database instances or database servers that offer an additional degree of freedom for distributing the load.

9.5.3 Application design

Any performance optimization at the level of the IBM FileNet P8 Platform servers cannot remedy severe flaws in the application design or the data model of a custom implementation. The Performance Tuning Guide lists important recommendations to remember when designing individual applications. The Content Engine can effectively manage billions of objects. However, to ensure proper response times, the application must avoid critical queries or design patterns.

The Content Engine data model does not restrict the number of objects or subfolders that can be filed into a single folder. Retrieving all content for a given folder is a memory-intensive operation, if many objects are filed into this folder because the Content Engine stores the result set in memory to perform additional operations, such as access control. Therefore we recommend either limiting the number of objects (folders, documents, custom objects) that are filed into a single folder to a number of 200 maximum or ensuring, by application design, that retrievals to the content of folders that have to contain a much larger number of objects are properly filtered so that the retrieval does not return more than 100-200 hits.

Index skew: A single folder that contains a large number of objects (subfolders, documents, custom objects) is subject to cause index skew, as described in section 9.5.2, “Database” on page 315, because all objects have the same value for the indexed attribute tail_id in the containment relationship table.

The Content Engine API version 4.0 supports a paging parameter (continuable flag), which allows you to subsequently retrieve chunks of result sets. We recommend that you use this option carefully because it might negatively impact the performance. Let us assume a query that returns 10,000 objects. If this query is executed with the continuable flag set to true and a paging size of 50 objects, the database is still treated to retrieval all 10,000 objects and then sort the top 50 results for the first page. In such a situation, it is significantly faster to execute the query to the Content Engine using a `select top 50 ...` clause and turning the continuable flag off.

For performance reasons, we recommend cleaning up the Content Engine Audit Event table and the Process Engine Event Log tables on a regular basis. A best practice approach is to export all data that is related to work objects that were terminated and for content objects, which are linked to terminated workflows, if this data is needed for audit or compliance purposes and delete them from the corresponding tables afterwards. Again, the event-based architecture of the IBM FileNet P8 Platform helps to identify the correct objects for exporting and deleting.

9.6 Distributing an IBM FileNet P8 system

The modular nature of the IBM FileNet P8 Platform and its scalability are important benefits when designing ECM solutions. Using IBM FileNet P8 Platform you can install components of the complete system at different locations to address given infrastructure restrictions (networks) and to improve the overall performance that the users experience. Throughout this section, we discuss how to approach the design of a distributed IBM FileNet P8 system.

9.6.1 Geographically dispersed users

Large organizations must support content management and content-centric processes over a large variety of geographical locations.

IBM FileNet P8 Platform offers a lot of configuration options that support the design of a distributed ECM system, and we briefly highlight the most important architectural building in this section. Refer to the *IBM FileNet P8 Distributed Deployment White Paper*[2] for a detailed treatment of the best practices for distributing an IBM FileNet P8 system.

[2] Available on request

Centralized IBM FileNet P8 system

In some cases a central IBM FileNet P8 system that clients access from different locations might be a valid solution. Alternatively, desktop virtualization technologies, such as Citrix or Windows Terminal Services, can also help to make applications available in various branch offices without installing servers locally.

Figure 9-11 on page 319 illustrates the architecture of a central IBM FileNet P8 system. All components are hosted at the central site, and the client at the remote site uses a wide area network (WAN) connection to communicate with the Workplace/Workplace XT.

The benefit of this configuration is the short distance between all of the core components, namely the Workplace/Workplace XT, Content Engine, Process Engine, and the storage layer (file systems for File Stores and databases). Additionally, managing a centralized system is easier compared to dealing with a distributed installation and its added complexity, especially backing up the data in a distributed topology can become a challenge.

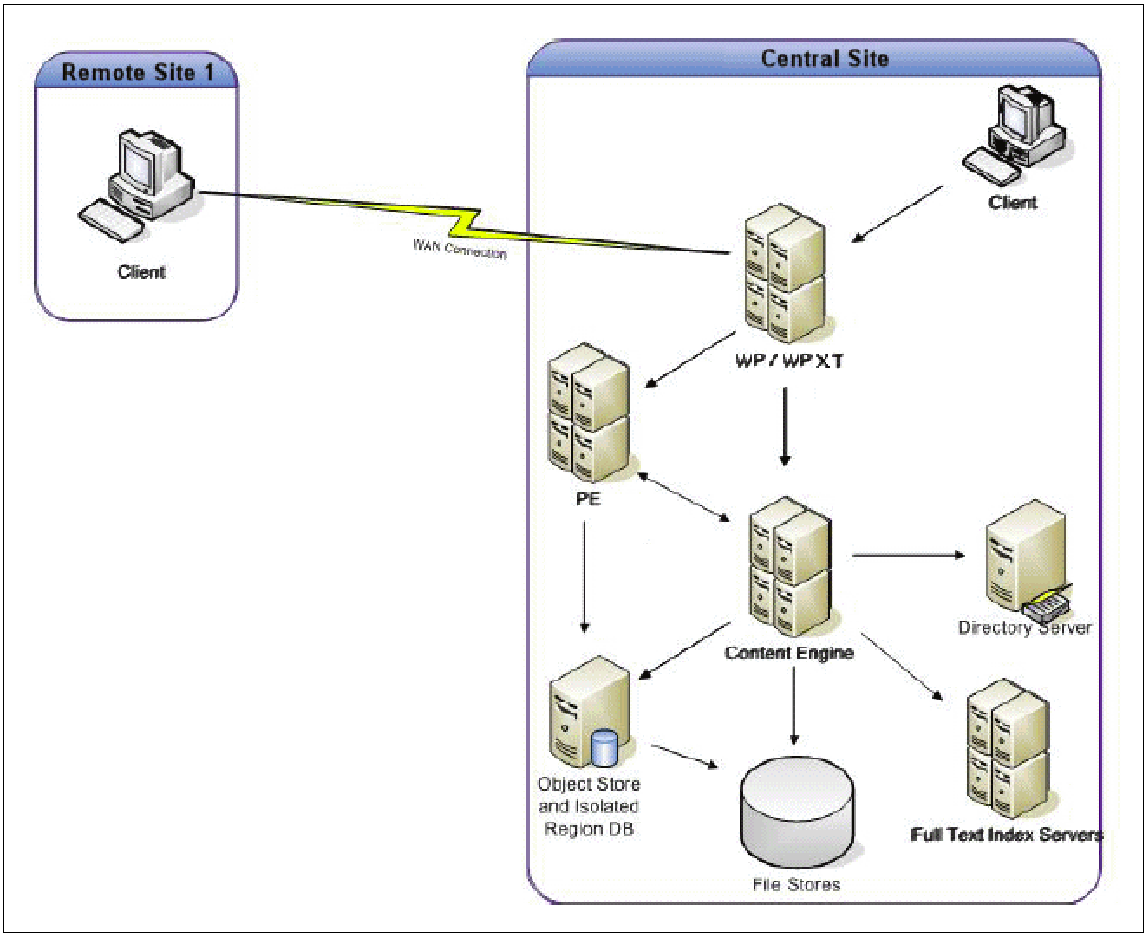


Figure 9-11 Central IBM FileNet P8 system

However, use cases remain that cannot be addressed by a centralized approach, for instance, in some countries legal rules exist that require that certain content must only be stored in the boundaries of the country itself. Another example is the work with large content objects, where a distributed storage is desirable to reduce the need to download the content from the central system to the remote location.

In general, we recommend starting with a central system and adding components at larger remote sites, if required.

Distributing Workplace/Workplace XT and Content Engine

The centralized approach can be easily extended by adding Workplace/Workplace XT and Content Engine servers at a remote location, which is illustrated in Figure 9-12.

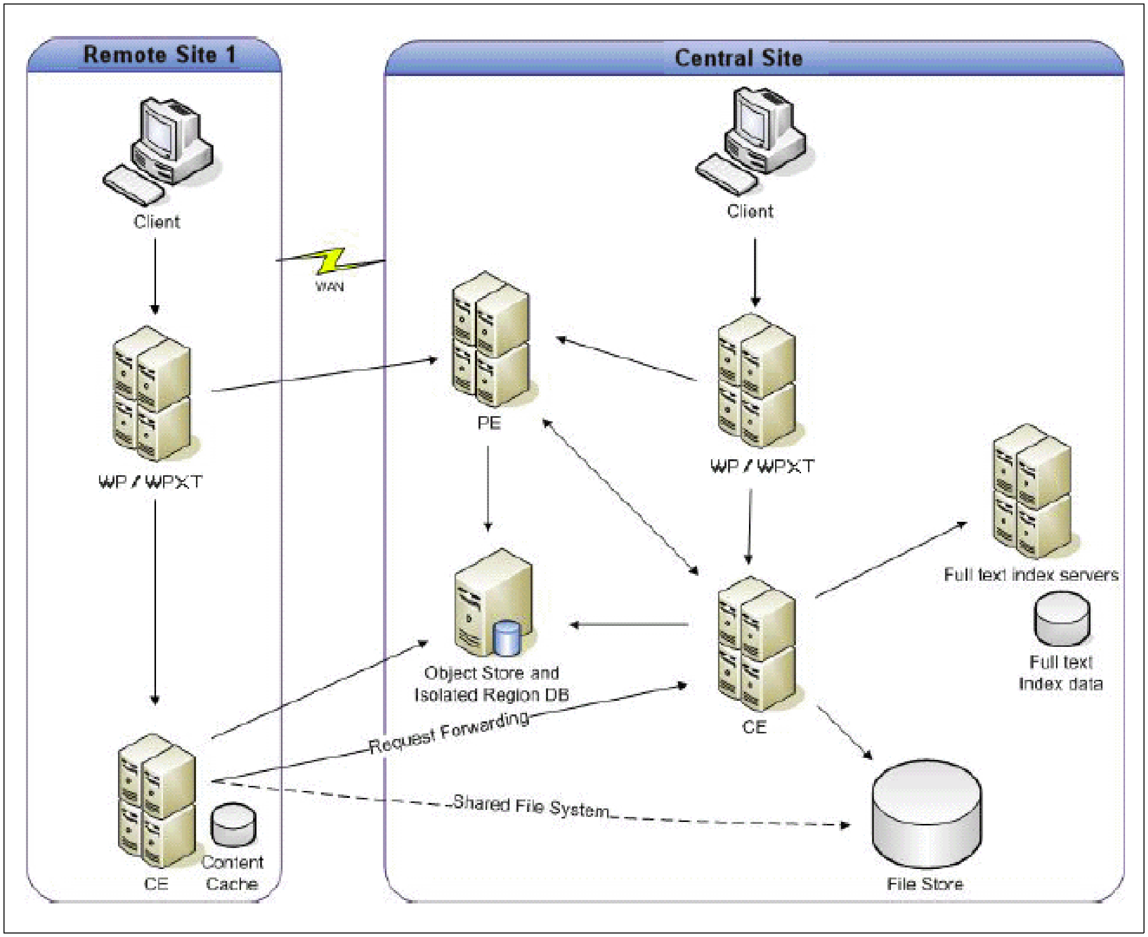


Figure 9-12 IBM FileNet P8 system with remote Workplace/Workplace XT and Content Engine

In Figure 9-12, the client at the remote site communicates with the Workplace at the same location. The Workplace connects to the local Content Engine and the remote Process Engine. This configuration benefits from the Content Engine cache so that content objects that are created at the remote site automatically remain in the local cache.

Content objects can also be preloaded to the cache at the remote site if they are created at the central site. For this purpose, either the event-based architecture of the Content Engine is used or the prefetch can be initiated from a workflow by using an appropriate Java component. The preferred method depends on the question, when the decision can be made, and if the content object is processed at the remote site and a prefetch is required.

In addition, a feature called request forwarding can be exploited to improve the performance at the remote site. Request forwarding allows the Content Engine that is at the remote location to forward queries to the central Content Engine for processing. The idea is that processing the request might include several round trips between the Content Engine and the object store database. By forwarding this request to the Content Engine that is located closely to the database server and only getting back the result set, there is less communication over the WAN connection.

Remote Workplace only

It is also an option to only deploy the Workplace at the remote site. In this configuration, the remote client benefits from a close proximity to the Workplace, but the downside is the WAN connection between the Workplace and the back end servers Content and Process Engine.

Because the Workplace does not provide content caching, content objects might travel several times to the remote location, for instance, if they are requested more than one time, which can be a problem, especially when working with large content objects.

Deploying only the Workplace at the remote site has advantages, mainly for applications where the client and the Workplace heavily communicate with each other and where latencies between the client and the Workplace are significantly reducing the response times that the users experience.

Process Engine

Distributing the Process Engine is significantly more complex because of the nature of data that this component processes. Process Engine work objects are typically much smaller in size compared to the content objects, so that limited bandwidth between the remote location and the Process Engine Server does not impact the performance too much. However, the effect of latencies on the WAN is still encountered. Furthermore, for content-centric processes, the work objects are not only processed by human participants, but there are also interactions with external systems that, in most cases, also were not distributed.

Also, take into account that the Process Engine's high transaction nature results in a large number of communications between the Process Engine Server and the Process Engine database because all work objects that are handled are managed in the database. Thus, a distributed Process Engine requires a Process Engine database at that location, too. It is important to understand, that the caching concept for Content Engine also stores the content objects locally only, not the metadata information, which is because the information in a database is subject to change more frequently than the content itself. For the Process Engine, this distinction cannot occur because the process instances that are managed are typically small in size and underlie frequent updates.

We recommend using a central Process Engine and distributed Application and Content Engines for most scenarios, as shown in Figure 9-12 on page 320. This topology benefits from the number of communications between the Workplace and the Content Engine towards the Process Engine. The Process Engine APIs are significantly less than the number of individual transactions that the Process Engine executes on the database.

Alternatively, if the process instances primarily remain in geographical regions, for example, when processing customer requests in North America and Europe, setting up different Process Engine systems for each region improves the performance, as illustrated in Figure 9-13 on page 323.

In Figure 9-13 on page 323, the master system is located in North America and consists of the Workplace (WP NA), Content Engine (CE NA), and Process Engine (PE NA). A database server at this location (DB NA) hosts the Content Engine GCD and object store databases for North America and the Process Engine database for North America. A file server (FS NA) provides the storage areas for content in this location.

The other system is located in Europe and also consists of the Workplace (WP EU), Content Engine (CE EU), and Process Engine (PE EU). Content is stored locally on a file server (FS EU), which either works as a pure content cache or also provides local file storage areas, depending on the detailed requirements. The process instances are managed in a local database (DB EU). European Object Stores are hosted on the database DB EU. However, the GCD information is always stored at one location only, in this case in North America. Access to content metadata for the remote location can benefit from request forwarding between the two Content Engines.

The downside of this approach is that work items cannot be transferred between both systems. Nevertheless, a client at one location can access the applications at both locations and can thus process the work items at the remote location (dotted line). However, when working with the remote Process Engine system, the access is over the WAN.

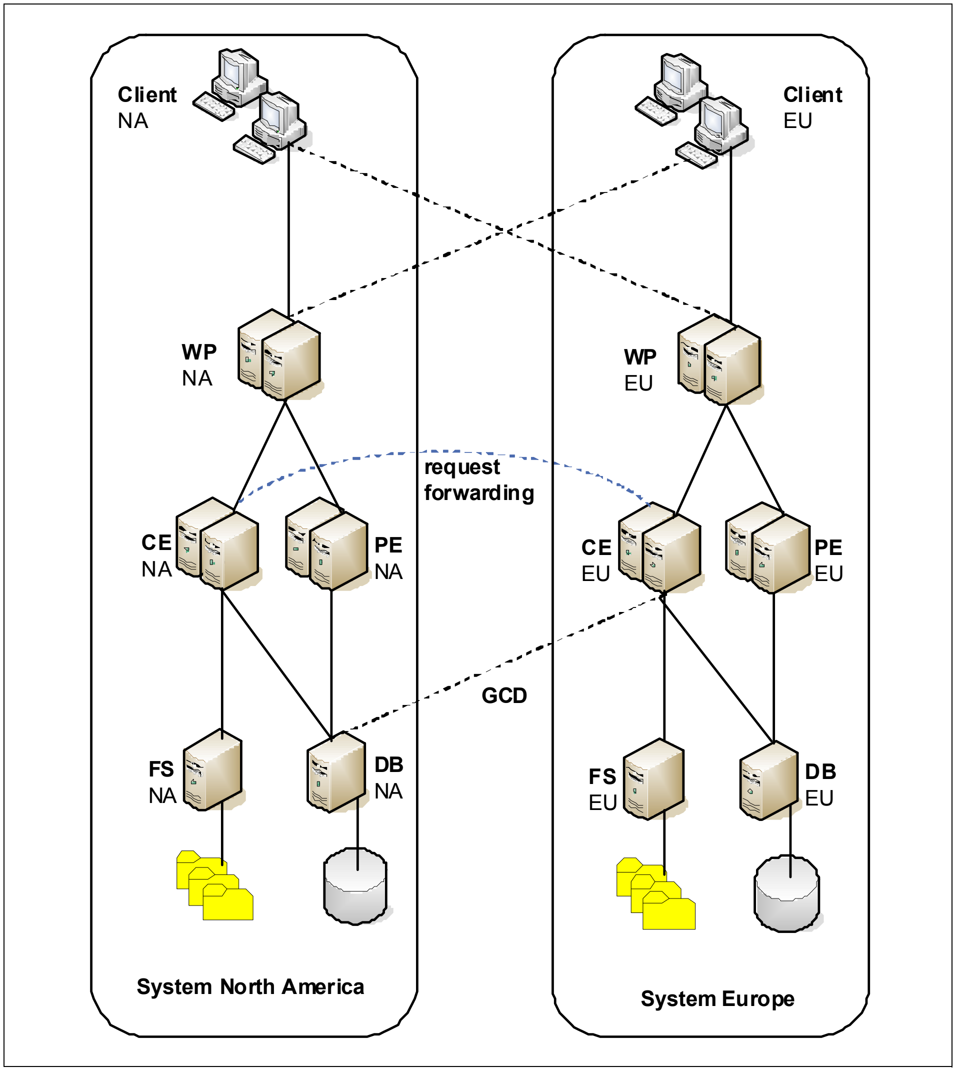


Figure 9-13 Distributed Process Engines

9.6.2 Disaster recovery

Disaster recovery (DR) addresses the loss of a complete data center or a complete site due to natural or man made disasters, such as fire, floods, or terrorist attacks. The goal of a disaster recovery strategy is to achieve business continuity with minimum interruption and data loss.

We do not want to go into the details of DR options for the IBM FileNet P8 Platform in this book because more details on this topic are discussed in the *IBM FileNet P8 Disaster Recovery Technical Note*[3]. We want to point out that disaster recovery and distributing IBM FileNet P8 systems are essentially two separate aspects. However, there are some interesting common aspects.

With the support of horizontal farming for all core components of the IBM FileNet P8 Platform, you might consider stretching the nodes server farms over a wider geographic area, for instance, in a metropolitan area network (MAN) with distances of up to 100 kilometers. Combined with mirroring the data layer (databases and file stores) such a configuration might theoretically address high availability and disaster recovery without the costs that are involved in implementing a dedicated disaster recovery infrastructure.

We do not recommend this simple approach because stretching the nodes of the farm introduces additional latencies for signals that travel over the wire and also introduce a new point-of-failure into the system topology, namely the MAN network. Typically this network is provided by an external service provider and is not as much inter control like the local area network (LAN), for instance, network lines might become interrupted by construction work, thus it will be required to implement redundancy by using multiple MAN connections, in the best case using different physical paths.

In addition, the MAN introduces additional latencies that are experienced in the communication between the different engines. We recommend that the distance between the two locations that are supposed to host the stretched farm be small (for example, 1-5 km) so that the impact of latencies is neglected, when considering the approach of a starched farm. On the other side, such a close proximity might not be able to address full disaster recovery needs.

As described in the Technical Note[3], the best practice approach for an IBM FileNet P8 system is to implement high availability using locally redundant components, such as a farm of servers, and to address disaster recovery by setting up a dedicated infrastructure at a secondary data center.

Note: By combining the capabilities of modern load balancing devices and a sophisticated configuration, it is possible to reuse the disaster recovery hardware, for example, as a test or QA platform, and reconfigure it in case a disaster occurs.

[3] ftp://ftp.software.ibm.com/software/data/cm/filenet/docs/p8doc/40x/p8_400_disaster_recovery.pdf

9.7 IBM FileNet P8 in a DMZ environment

There are several use cases where an IBM FileNet P8 system is accessed not only internally by persons that are part of the organization, but also externally, for example by business partners or by customers, especially deployments that use business process management might incorporate access from external participants, for instance, a claim process might be started by a customer using a web based self service portal. The customer is supposed to be able to track the progress of his or her cases based on a portal application. External experts can be part of the process, and they might upload content, such as pictures of the damage or reports into the case management system. Typically, organizations allow access for external participants to their internal systems only through a demilitarized zone (DMZ).

As described in Table 9-1 on page 278, a DMZ is typically an area that is secured and separated from the outside word (the Internet) and from the internal network by two different firewall systems. The aim is to reduce the risk that intruders from the Internet can get unauthorized access to the internal IT systems. To achieve the optimum protection, the protocols and ports that the firewalls route are restricted as much as possible.

Figure 9-14 on page 326 illustrates the general configuration for a DMZ with two firewalls.

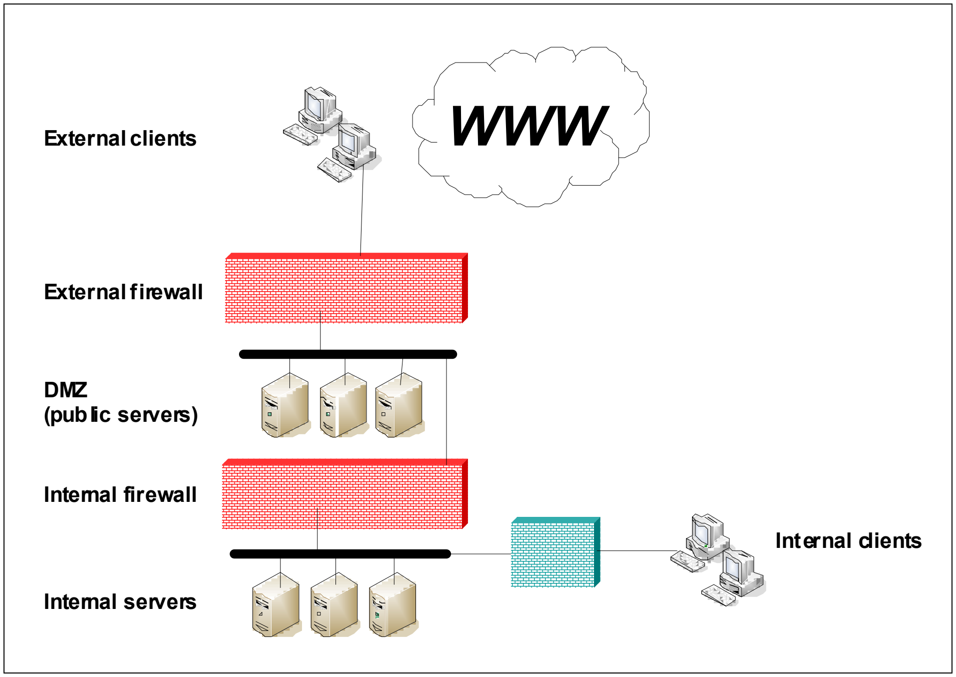


Figure 9-14 DMZ with dual firewalls

The internal clients can access the internal servers directly or through a separate firewall.

Web server with http proxy in the DMZ

The best practice approach for a DMZ deployment is to implement a clear separation between the IBM FileNet P8 servers in the internal network and the external clients outside of the DMZ by web servers with http proxy capabilities. This reverse proxy must be transparent for both the external clients and for the IBM FileNet P8 servers.

In a typical scenario, the web servers in the DMZ host some (in most cases) static external web applications and also perform some sort of load balancing in the way that they distribute incoming request to the internal servers. These web servers are typically not only used by the IBM FileNet P8 applications, but also by other applications that need access from external clients. The web servers are configured to forward incoming requests based on the URL to the appropriate internal servers. Figure 9-15 on page 327 illustrates this scenario.

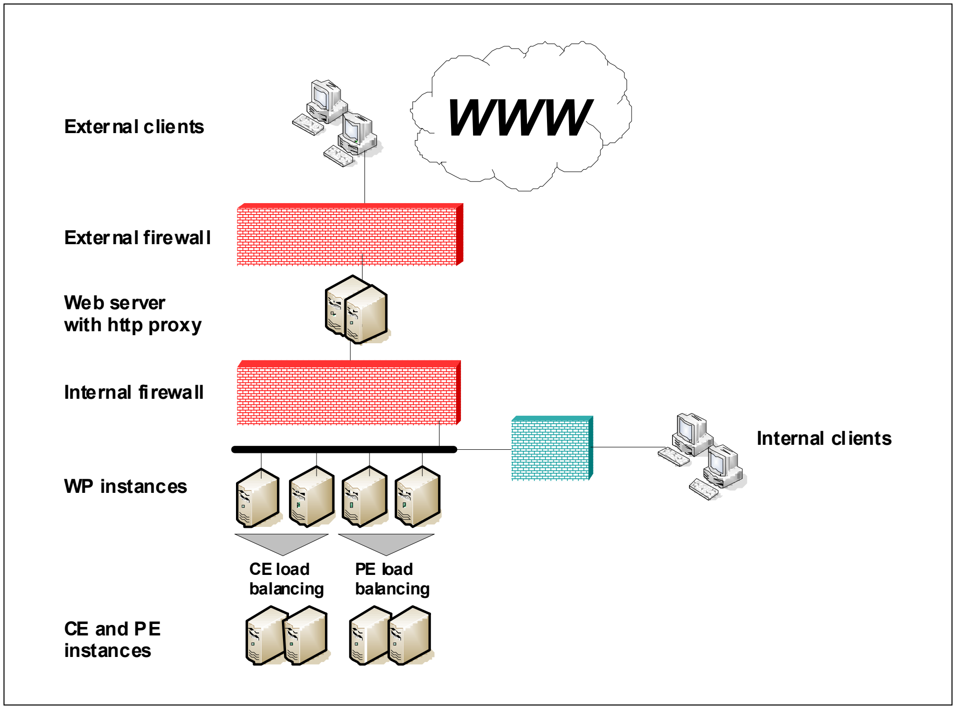


Figure 9-15 IBM FileNet P8 DMZ deployment best practice

For the external clients, the reverse proxy acts as the application server, and for the application server, the reverse proxy acts as the client. Therefore the reverse proxy must rewrite any packets that come from the external client in a way that they seem to originate from the reverse proxy server instead, which ensures that the application server passes the response to the reverse proxy server instead of trying to directly contact the client. Conversely, the reverse proxy must rewrite any response that it receives from the application server in a way that it seems to originate from the reverse proxy. This setup ensures that the client who evaluates the response reconnects to the reverse proxy for subsequent requests instead of directly trying to access the application server, which fail because the application server is behind the internal firewall.

Workplace in the DMZ

Under certain conditions, it is beneficial to put the Workplace for external users into the DMZ. In this scenario, the Workplace, or a farm of Workplace servers, are installed at the level of the DMZ to serve the clients in the external network. Typically, there is another set of Workplace Servers that run in the internal network that serve the internal clients and that also run the Component Manager instances. Figure 9-16 on page 328 illustrates this configuration. The other internal servers, such as Content Engine and Process Engine, are not shown.

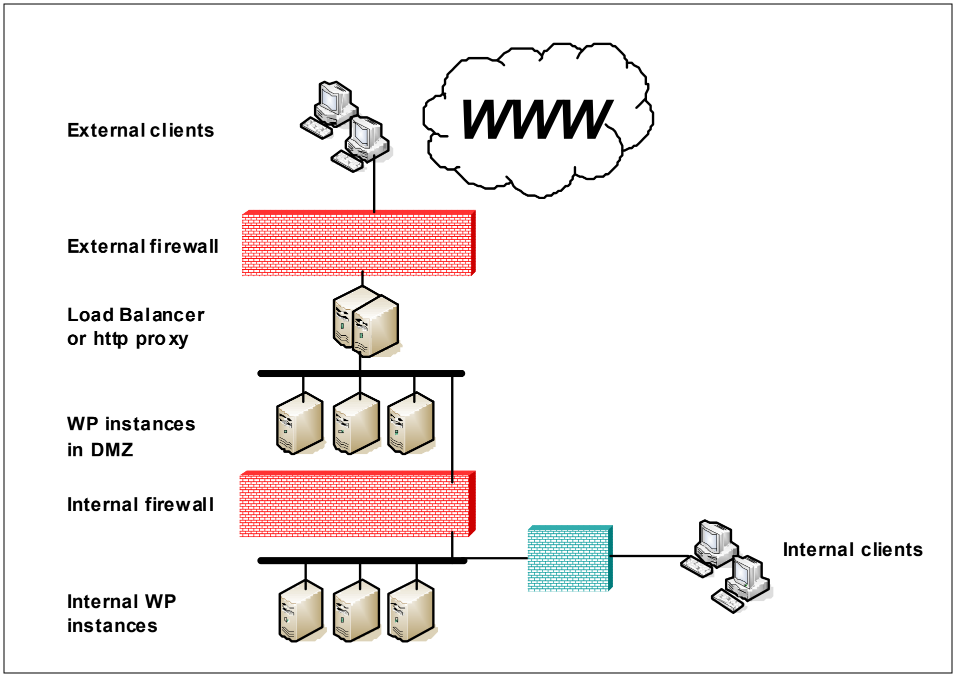


Figure 9-16 Workplace (WP) in the DMZ

This architecture, in Figure 9-16, delivers the benefit that clients can directly access the Workplace, which can be mandatory, for example, if HTTPS communication between the client and the Workplace is intended to be used for external clients, and the HTTPS protocol cannot be routed across the DMZ.

9.8 Sample deployment

When designing an IBM FileNet P8 system, architects are not limited to exclusively adhering to only one of the concepts that we mentioned in 9.4.2, "Content Engine" on page 294, for scaling the Content Engine and managing the workload. Instead, in most cases we see a combination of different approaches to cover the full spectrum of applications. Typically, this involves application server work management for applications using the Java API and the EJB transport and hardware load balancing or an http plug-in for web services clients.

In this section, for a sample deployment, we describe how virtualization can be used to effectively take advantage of hardware resources for a shared platform.

We assume that the following requirements are given for the sample deployment:

- Implementation of a shared infrastructure that serves multiple IBM FileNet P8 based projects.
- Template-style approach for using cloning to create the environment for a new application.
- Implementation of high availability for each application.
- Use AIX as the operating system, and limit the number of LPARs.

Based on the requirements, the following system architecture is derived:

- For each application, a separate set of the three IBM FileNet P8 core engines are installed into one separate LPAR.
- Use another LPAR for each application that runs the database and the storage for this application.
- Use farming for the IBM FileNet P8 core engines (Workplace, Content Engine, Process Engine) to ensure high availability.
- Use a traditional active passive HA cluster for the database and the NFS server (refer to the IBM FileNet Content Manager Implementation Best Practices IBM Redbooks publication[4] for a detail description of HA clusters).
- Use two separate IBM p570 servers to achieve redundancy at hardware level and distribute the nodes of each farm across both servers.
- Use a set of two redundant hardware load balancers to distribute the load across the Workplace and Process Engine farms and for the Content Engine WSI transport.
- Use application server-based work management for the Content Engine farms for the EJB transport.

Figure 9-17 on page 330 illustrates the basic idea of installing an IBM FileNet P8 domain into two LPARs. The LPAR named “LPAR application” hosts the IBM FileNet P8 core engines and is accompanied by a second LPAR (LPAR storage) on the same physical machine that provides the database and content storage.

[4] *IBM FileNet Content Manager Implementation Best Practices and Recommendations*, SG24-7547

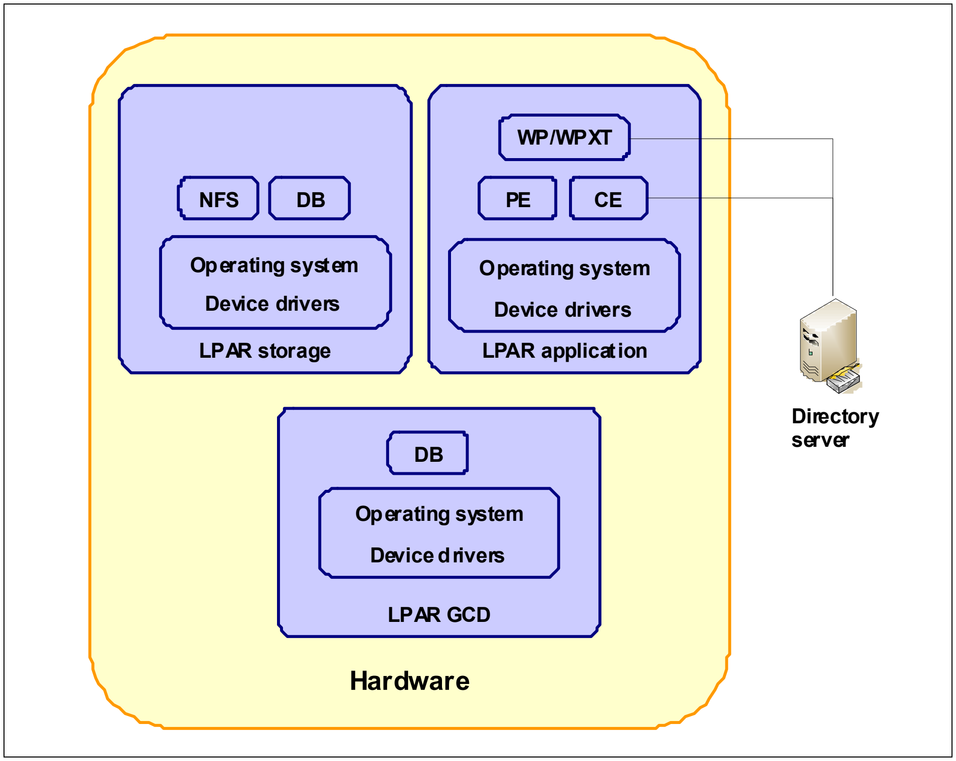


Figure 9-17 Design pattern for a clonable IBM FileNet P8 domain

The configuration in Figure 9-17 delivers the benefit of having building blocks that are fairly easy to clone to set up the environment for a new application. One central LPAR is used to host the GCD database for the domain.

By enhancing this approach with another LPAR application on the second server that hosts a second set of the IBM FileNet P8 core engines and configuring them as a farm for Application, Content Engine, and Process Engine, the requirement for high availability can be addressed on the application and engine level.

Introducing another LPAR storage on the second server and implementing the database and NFS server as HA clusters across both storage LPARs ensures that high availability is also considered at the level of the storage layer.

Figure 9-18 on page 331 illustrates these architectural considerations for an example of three different applications that run on the two servers (Server A and Server B) and are fully segregated from each other.

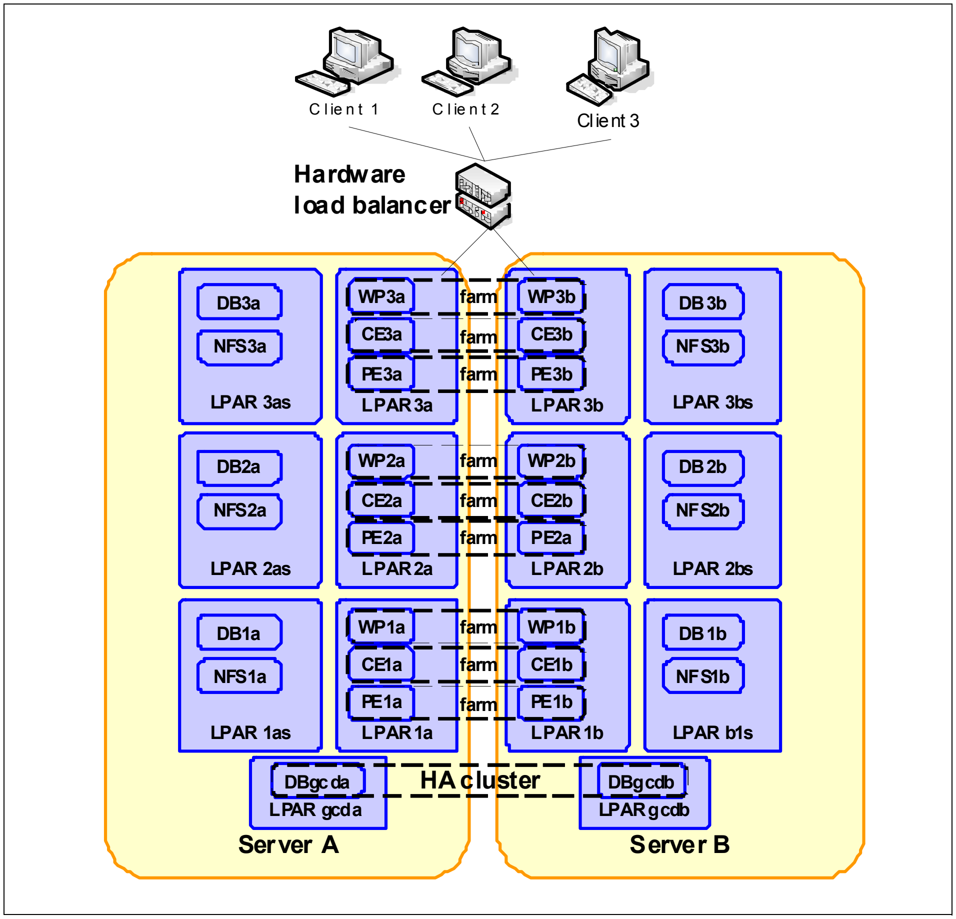


Figure 9-18 Virtualization used to implement high availability on a shared infrastructure

The application that is used by Client 1 uses the farms that are established by WP1a/WP1b, CE1a/CE1b and PE1a/PE1b. Client 2 uses WP2a/WP2b and so forth. In Figure 9-18, only the hardware load balancer is explicitly drawn. For the Content Engine Java API clients that run the EJB protocol, application server based load balancing is implemented across the Content Engine farms. Also, the HA clustering is only shown explicitly for the GCD database, even though it is also applied to any database that the applications use and for the servers that provide the NFS shares for the applications.

Refer to the solution templates that we describe in Chapter 10, “Architecting an IBM FileNet P8 solution” on page 333 for more information about typical IBM FileNet P8 deployments.

10

Architecting an IBM FileNet P8 solution

In this chapter, the features of the IBM FileNet P8 reference architecture are applied to a solution illustrating the process of FileNet solution design. The process consists of breaking down a business problem into a set of requirements and fulfilling these requirements using IBM FileNet P8 components and services. The solution is finalized by aligning elements of the solution to non-functional requirements.

This chapter covers the following topics:

- 10.1, “Basic approach” on page 334
- 10.2, “Solution overview” on page 341
- 10.3, “Solution template: Customer services support” on page 342
- 10.4, “Solution template: Enterprise-wide document management” on page 349

Disclaimer: The scenarios that we describe in this chapter are fictitious. We provide them here for reference purposes only.

10.1 Basic approach

The basic approach for designing an IBM FileNet P8 solution architecture is to break up the business needs into functional requirements and identify which modules fulfill those requirements. The modules have requirements of their own, such as which operating systems or application servers are supported. These are non-functional requirements.

It is essential that the following solutions are used only as examples and be considered as a general methodology for developing a solution. Real world projects involve many factors that require a detailed analysis before a solution can be developed.

For architectural considerations, functional requirements must be fulfilled. Besides the capabilities needed by users to fulfill their job, there are other specifications to be met. Non-functional requirements describe the quality of the future solution and constraints that cannot be changed within the scope and lifetime of the project. When designing a solution based on a dedicated specification, some general questions must be answered:

- What are the functional business requirements?
- Which product modules fulfill the business requirements?
- Are there any leading architectural mandates in addition to the functional requirements?:
  - Legal and IT Governance requirements.
  - Fulfillment of requirements based on corporate strategic directions.
  - Will this environment the intended for production or development use?
- What are the availability requirements?
- What are the performance requirements with respect to respond times and throughput?
- How many environments have to be built?
- How are project costs funded? Is there and emphasis for keeping project budget low or are costs based on total cost of ownership (TCO)?

10.1.1 Module selection

Architectural decisions hinge on the selection of the product modules corresponding to functional requirements. Each module has dependencies independent of the other modules and as such their selection influences architectural requirements. The following sections describe a a fictitious set of requirements and how they align with product capabilities.

10.1.2 Leading architectural requirements

After having selected adequate modules, it is necessary to consider requirements that are not necessarily business requirements. We can divide these requirements into three groups:

Leading requirements generally fall into one or more of the following groups:

- Legal and IT Governance

  All organizations are subject to local laws and most organizations are subject to some degree of IT governance. Sarbanes-Oxley (SOX), for example, requires American companies who publicly offer stock to declare controls ensuring executive accountability. These controls typically refer to security of information relating to the reporting of earnings. Another example is the Health Insurance Portability and Accountability Act (HIPAA), which contains language that is intended to protect the privacy of an individual's health records. Additional requirements might be mandated by design standards, such as the Standards and Architectures for eGovernment Applications (SAGA) or service delivery frameworks such as Information Technology Infrastructure Library (ITIL). The fulfillment of these requirements can demand architectural considerations over and above the business requirements.

- Strategic directions

  Nature of a corporate directions are general guidelines concerning operating systems, application servers, databases, virtualization, and many other (technical) regulations without reacting to dedicated requirements but to demand standardization within the environments. Conflicts with functional requirements must be identified at an early stage of the architectural design because exceptions might cause other dependencies to be followed.

- Requirements based on the type of environment used

  An IBM FileNet P8 solution architecture usually includes more than one environment to ensure that there is adequate infrastructure available to run the application in production and have that environment be available for other work. Additional environments might be needed for development or quality assurance purposes and might have needs that are different from the production environment. Specifications for surrogate environments must be developed while maintaining as much similarity to the production environment as is practical. Table 10-1 on page 336 shows examples concerning various environments and their usual specifications (exemplary specifications, at least three dedicated and independent environments are recommended: development, test, and production environment)

Table 10-1 Sample environments and their typical specification and characteristics

Environment	Usual specification	Characteristic
Production environment	► Functional and non-functional requirements must be fully met ► Software and configuration changes must be validated in lower environments	► Highest performance and availability requirements ► Virtualization where applicable
System or integration test environment	► Product modules are equal to the production environment ► Physical implementation must be comparable with the production environment ► Can usually combine with other test environments	► Highly virtualized as long as performance requirements do not prevent virtualization ► Performance requirements are medium high (integration test requirements lead the performance requirements)

Environment	Usual specification	Characteristic
Disaster and recovery test environment	► Product modules are equal to the production environment ► Physical implementation must be comparable or equal with the production environment ► Can usually combine with the environment for performance tests	► Use cases of the disaster and recovery tests conduct the characteristics of the environment ► If the focus is on functional test cases, virtualization is a valid option for the environment ► If the test cases must demonstrate recovery times or other proofs with technical dependencies, equal or comparable (to the production environment) hardware and virtualized systems must be designed
Performance test environment	► Product modules are equal to the production environment ► Physical implementation must be identical to the production environment or at least must be able to be interpolated ► Can usually combine with the environment for disaster and recovery tests	► Performance requirements are identical to the production environment (or at least comparable) ► Virtualization equals the production environment

Environment	Usual specification	Characteristic
Education environment	► Product modules are equal to the production environment ► Small number of servers ► Some functionality might be stubbed	► Highly virtualized ► Performance requirements follow the number of concurrent training participants ► Clustering must be implemented if used in the production environment (not a must if higher environments beneath production environment are available)
Development environment	► Product modules are equal to the production environment except where appropriate modules for high availability ► Small number of servers ► Some functionality might be stubbed ► Collocation of modules might differ from production environment	► Highly virtualized ► Lowest performance requirements ► Clustering must be implemented if used in production environment for development purposes (not a must if higher environments beneath production environment are available)

10.1.3 Availability requirements

The purpose of collecting availability requirements is to make investments in strategies that minimize the operational loss of a system should the system be unavailable for business use. Requirements concerning the availability of a solution might follow a grading system that categorizes the level of importance the application represents to the business or maybe based on a more direct weighting of unrest against the cost of a countermeasure.

Availability is not one particular technology but rather a collection of strategies. Each countermeasure is incrementally more costly to implement than the next that provides additional availability. The most basic form of availability protection is a system back up, where application binaries and business data is copied to a system other than that used for production. This is followed by system hardening or multiple redundant components that are employed to reduce the number of single points of failure. High availability builds upon the concept by having multiple redundant servers take over for failed servers. Lastly, disaster recovery protects against the loss of the data center by duplicating an entire solution at an alternate geographic location.

The availability of an IBM FileNet P8 solution is dependent upon the availability of the communication paths, which fulfill the business requirements. While sometimes this represents the whole solution, in practice some components are less critical than others, for example, an organization that only rarely uses Content Search Services might not weigh components as heavily as the Content Engine. The risk to the business choice of countermeasure must be weighted to consider the implementation and maintenance costs, follow-up costs against the risk of unrealized revenue, and lost worker productivity.

The availability of the solution is determined by its weakest link. Each module might have separate availability requirements because of the design of the software or because of the business requirement it fulfills. Deriving the proper strategy can only be determined by weighing the risk, the cost of the risk, and the cost of the countermeasure. Availability requirements are usually defined by service level agreements (SLA).

Further information about availability is provided by the following RedBooks:

- *IBM High Availability Solution for IBM FileNet P8 Systems*, SG24-7700
- *Disaster Recovery and Backup Solutions for IBM FileNet P8*, SG24-7744

10.1.4 Performance requirements

A system specification or an SLA might also define performance requirements. Performance specifications define minimum requirements, response time, and throughput.

The core IBM FileNet components (Workplace/Workplace XT, Content Engine, Process Engine) support horizontal scalability where multiple systems providing the same service work in tandem, minimize response times, and maximize throughput performance. See also 9.3.1, “Horizontal scaling: Scale out” on page 280.

Solution performance is limited by the lowest performing component along a particular communication path. All computer systems are implemented as layers, for example, the performance of checking in a document, a function of Content Engine, is not based on the performance of the Content Engine alone. The system that implements the Content Engine API, the network between the client and the Content Engine server, the Content Engine software, the application server software, implementation of the Java Virtual Machine, the network between the Content Engine server and the NAS, the network between the database and the Content Engine, the database software, the operating system on which the database software runs, the operating system of the NAS, and storage area network between the NAS and disk, all play a role in the performance of what, at the surface, seems to be a simple operation. Performance management, evaluation, and optimization for the purposes of requirements gathering must consider every component through which the data flows.

10.1.5 Number of environments

Besides the previously named conceptual formulation of a dedicated environment, the number of environments to be implemented must influence the architecture of a certain environment. Dependencies originate from shared resources and limitations of interference by third-party systems (for example, the number of FileNet environments is greater than the number of corresponding external applications).

10.1.6 Total cost of ownership

Functional requirements and cost directions have important impact on the architecture. The following competing conditions must be equilibrated:

- SLAs have to be effectively met.
- Risks of hardware failures and resulted system standstills, times of unavailability and derived costs of downtimes (labor times, sales shortfalls) have to be regarded.
- In case of systems, that shall be expanded capacities and their physical implementation phases have to be decided.
- For any question and its answer usually more than one option exist with different costs.

At any time, additional costs must be evaluated versus the additional value of the related solution. Maintenance costs of the system and the dedicated implementation costs must be put into relation to each other.

10.2 Solution overview

To help you architect an IBM FileNet P8 solution, we provide solution templates that show how some IBM FileNet P8 implementations are deployed. IBM FileNet P8 implementation can grow over time, so the solutions that we present in this chapter are in the order such that growth over time is reflected because the architecture diagrams change between solutions.

We provide the following solution templates:

- Customer Services Support

  Single-site implementation. It has low volumes, has content, process, and case management. No high availability involved.

- Enterprise-wide Document Management

  A worldwide distributed system for all documents in the customer facing and Human Resources departments. Heavy business process and content management use. Multiple offices distributed across the world's regions. This implementation uses scanning, electronic forms, and productivity suites that are integrated with the IBM FileNet P8 Platform to provide content creation.

Each solution template represents various areas to which the IBM FileNet P8 architecture can be applied, with differing sizes, user interfaces, and integration points, which allows us to discuss the specific points of business value and explain architectural decisions better in a practical context. The solution templates discussed are based on real-life, live IBM FileNet P8 implementations.

For each solution template, we present the type of deployment, either small or large, based on our previous customer experiences. We add a mock business scenario around it to provide explanation as to why we chose to make certain solution and architectural decisions in the design. We also list the particular business problems to be solved and how features of the IBM FileNet P8 Platform solve the problems and provide business value over and above what the customer originally envisioned.

Each solution template consists of multiple sections. The solution overview section provides high-level information that is designed to address the main business problems and identify products to be included and platform features used. As an extension of this, the future enhancements section goes beyond what is needed as part of the core solution to explore additional enhancement that can be made to drive additional value from an IBM FileNet P8 Platform implementation.

The bulk of the information is included in the solution architecture. The architecture section addresses customer and solution-specific issues that require the IBM FileNet P8 Platform to be designed in a particular way. These issues can be constraints, such as multiple sites, existing corporate software, IBM FileNet P8 products used, and sizing.

In addition to the purely IBM FileNet P8 architecture information, we also included, where applicable, information about existing industry process and data models. IBM FileNet P8 has direct mappings into the more content-centric processes and information stores of many of these models. This information provides a greater background to explain how an IBM FileNet P8 solution can fit into an organization's IT infrastructure.

10.3 Solution template: Customer services support

This is a single-site implementation without high-availability setup. It has low volumes of small documents (letters and eForms) arriving, each with a management processes being launched. The solution requires content, process, and case-management technology. Processes are relatively long lived to approximately 30 days.

10.3.1 Scenario

A water and gas utilities company is finding it hard to fix broken pipes and respond to customer requests. This difficulty is partially due to the number of customer requests, the inconsistent process being applied by various staff of different levels of experience, and inefficient paper-based processes.

10.3.2 Business problems and their solutions

Based on the business scenario, the company has the following business problems:

- Inconsistent processing
- Customer Service Level Agreements not being met
- Hard to find documents and lost documents
- Storage costs increasing
- Costly customer servicing

Inconsistent processing

Inconsistent processes lead to missing information and incorrect assumptions and decisions being made, which cost time to fix because technical staff must second guess what the customer originally wanted; consequently, the technical staff usually does not have access to the original information.

Solution

Using IBM FileNet P8 *active content* technology, automatically initiate the correct complaint-handling process for a particular or general problem area, which removes initial manual handling and routing of the complaint.

Build the complaint handling process using IBM FileNet Business Process Manager, which makes the process well defined and removes the chance of it being carried out inconsistently.

Customer Service Level Agreements not being met

The company is in a regulated industry and is subject to heavy fines if customer service requests are not handled in a timely manner.

Solution

When building the complaint handling process, configure timers to escalate work based on Customer Service Level Agreement (SLA) targets. Use IBM FileNet P8 Advanced Case Management (ACM) to manage work items and automatically prioritize work within an inbasket. Using this solution the company can merge cases where the same problem was reported by multiple people, which increases the information that we have about the reported problem and increases the speed in responding to it.

Hard to find documents and lost documents

Many customer requests are mislaid in a large area that is used for document storage. Also, files are often sent to people to process and are unavailable to others when needed or when on holiday.

Solution

Use Electronic Forms (eForms) to enable customers to report issues online. This solution ensures that the maximum useful information is instantly recorded rather than waiting for paper to arrive. This electronic information ensures that the information is accessible to any permitted personnel at any time and that it is not being lost somewhere. It also means less paper to handle.

Storage costs increasing

The cost of storage increases due to more complaints.

Solution

Remove paper out of the company by introducing bulk scanning with IBM Datacap into an enterprise content management repository. Extract easy-to-identify data, such as customer name, number, account number, date of report, and customer address, which makes instant savings of cost in searching for data and paper storage. We are assuming that most of the documents will have a size not larger than 150KB and sometimes will reach 1MB. So there will be no need of a dedicated file store because all documents are stored within the database.

Costly customer servicing

Customers who phone in to ask about the status of their requests are hard to service, and the calls are often long because of the manual nature of the current work environment, which leads to high customer service costs.

Solution

Steps within the complaint handling process are configured to proactively inform customers of the status of their complaints, thus reducing the likelihood of them needing to call the customer services team and greatly improving efficiencies.

Using filters in IBM FileNet P8 ACM show active cases that match certain criteria. When customers phone in, it is easy to find their reports by searching by customer account number or area code. Also make customers' status available online such that they can look up the status on their own.

10.3.3 Customer architectural constraints

The company has about a hundred central staff located in a single office. There are currently 2,000 letters coming in per day, Monday to Friday, arriving at 8:00 AM that need scanning as soon as possible for the workers to start handling them on the same day. This situation requires a high speed, bulk scanner and content repository that can handle high-peak ingestion rates and a constant rate of document retrieval during a day.

Integration with external systems requirement is to post code verification and email systems and a SMS text message web service.

The company's current operating environment is Windows with a Microsoft SQL Server database and Network Attached storage exposed as a CIFS share. The company is also keen to use existing IBM hardware saved over from a previous project. No disaster recovery or high availability is required.

The company has tight time lines and is keeps implementation costs low. The internal users will use browsers to process work. The external customers are also expected to use browsers to check their status. Therefore, no client-installed applications are considered.

10.3.4 Solution architecture

Figure 10-1 shows the target single-site environment for this IBM FileNet P8 solution. Because of the relatively low demand in processing volumes and speed (per our sizing tool calculation, which we do not cover in this book), the Content Engine and Process Engine can be consolidated into one single server and is still suitable at peak times.

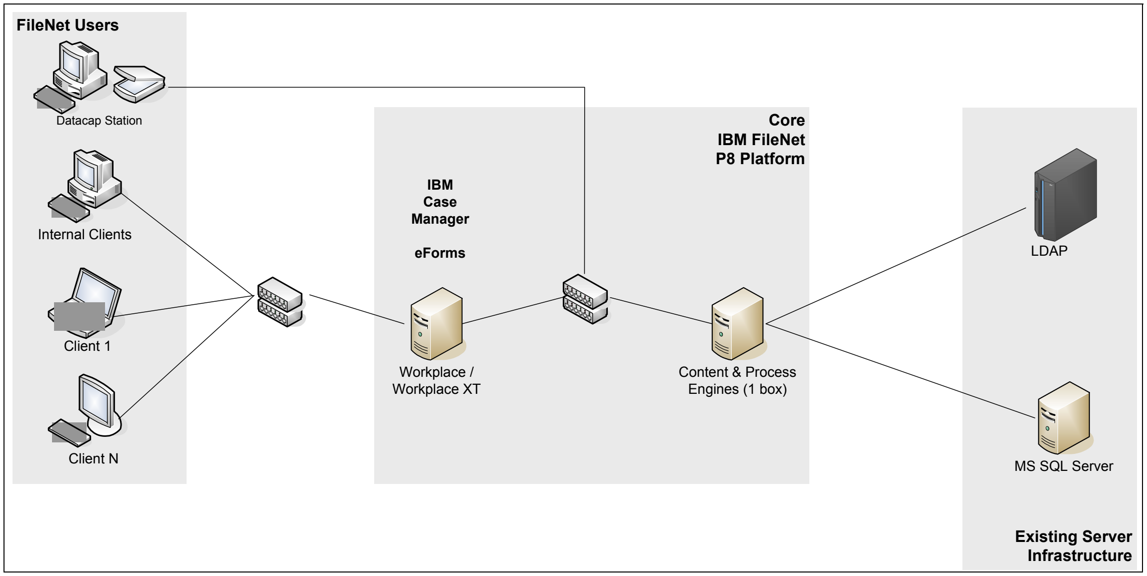


Figure 10-1 Core solution architecture[1]

Target architecture assumptions

By adding eForms as a separate channel, we assume 40% of all future complaints enter the system through the customer or call center staff completing an eForm.

Systems hardware

The Content Engine, Process Engine, and Workplace/Workplace XT, MS SQL Server, and Directory Server (are all on IBM System x3500 M3 (Windows) machines).

[1] IBM Case Manager requires Workplace XT and is not available in Workplace only environments.

Sizing consideration

Although we do not cover the sizing of the system, we discuss some sizing-related considerations:

- Consider how long you will retain the content. We assume a content population of five years of operation to be retained.
- Be aware that the Content database grows large overtime because we want to record one case object for every document and the document itself. In reality, cases can be merged with the correspondences for multiple complaints that report the same issue. So the overall case number is less.
- Consider whether to keep the case history audit fields for all time.
- Think about the information life cycle in any system to avoid creating a digital landfill of information that you never intend to use.
- Consider the additional processing demand when using eForms. By having eForms as an input mechanism, it increases load on the application server because the form is rendered one time for filling in and again later for review. This is different from incoming letters because scanning the letters includes a create document operation, rather than both a create document and render form operations as for eForms.

10.3.5 Solution processes

For this solution, we create four processes to handle the workload:

- Generic Customer Request Handling process

  This is launched when we cannot automatically determine the type of complaint. The job goes to an unsigned work queue for someone to manually assign the type of complaint that is used. The relevant process is then launched.
- Register Complaint process

  When we know the customer request is a complaint, launch this process to extract complaint information and assign to the correct department. This automatically launches the Works Required process.
- Works Required process

  This handles collation of similar reports and cases, assigning a works team, inspection, and correspondence with the reporting customers.
- Incoming Additional Correspondence process

  The Works Required process might wait for additional customer correspondence. This process handles additional correspondence documentation, files it against the appropriate case, and reawakens the main handling process.

As an example, Figure 10-2 shows the details of the Works Required process. In this process, we create a case with all of the relevant reports and process the case. For the processing case step, we assign a manager for the case, set the appropriate security, and service the client.

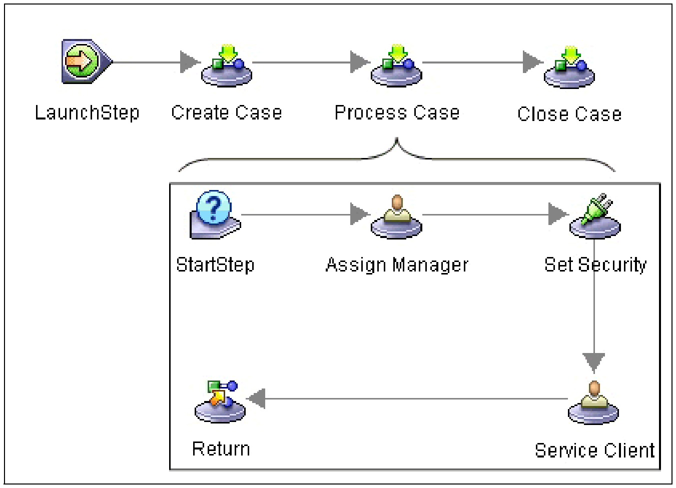


Figure 10-2 Works Required process

10.3.6 Future enhancements

For future improvements, the company can optionally do the following enhancements:

- Configure IBM FileNet Process Simulator and IBM FileNet Case Analyzer to identify process road blocks based on current and future staffing levels. Use this in the future to improve the process and make it more efficient. Also use the gathered information for business intelligence and reporting.
- Use IBM Cognos Real Time Monitoring to provide graphical feedback to managers of work at varying stages and within or out of SLA targets. Also aggregate data by location to give a pictorial view of where problems occur.
- Use IBM Classification Module to determine the type of problem being reported, for example, a leak, incorrect bill, or loss of water to street. This feature makes deciding which team to send the customer complaint to be more automated, which increases work processing throughput.
- Integrate the business process with an existing customer information or CRM system. This solution gives process workers access to customer contact

details other than those that are provided during reporting. This solution also enables the system to spot when details change so it can initiate a Customer Details Validation and Modification process to make the internal data more accurate. Storing customer contact preferences means that your customer notifications can be sent through their preferred channel (phone, email, SMS, or post).

If all of the above components are added to the final solution, the core functionality is kept as it currently with the extra components shown in Figure 10-3 on page 349 added.

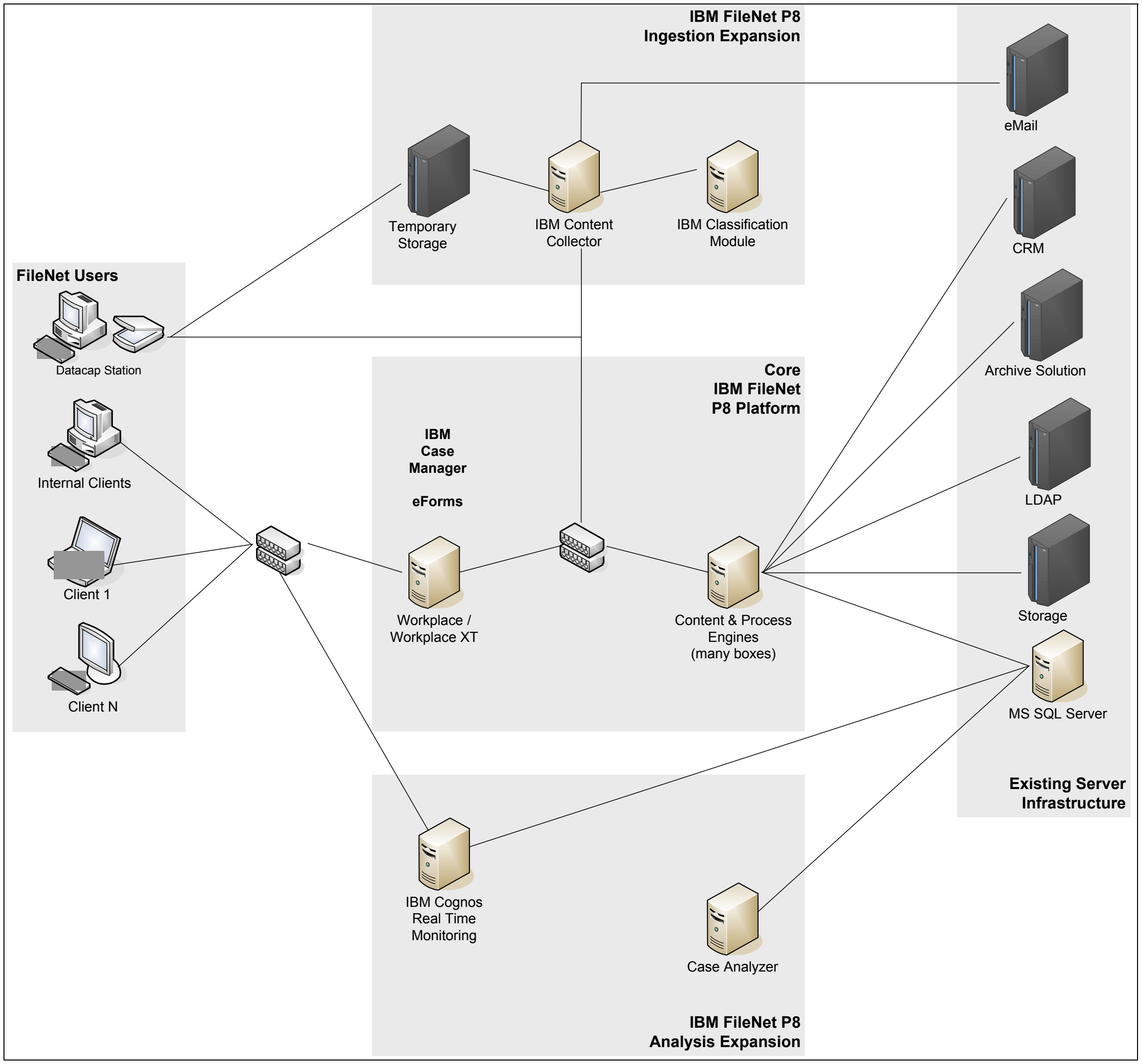


Figure 10-3 Solution with all optional components included[2]

10.4 Solution template: Enterprise-wide document management

This section discusses a worldwide distributed solution for all documents in the customer facing and Human Resources departments. The implementation

[2] IBM Case Manager requires Workplace XT and is not available in Workplace only environments

involves heavy business process and content management use. There are multiple offices distributed across the world regions that use scanning, electronic forms, and productivity suites integrated with the IBM FileNet P8 Platform to provide content creation.

10.4.1 Scenario

The company is a multi-national company with offices and regional hubs around the world. They have a long history of using enterprise content management systems as departmental or country-wide solutions. They now want to move towards completely eliminating paper for all internal business functions.

This change involves capturing all customer-related documents, internal Human Resources information, policies and procedures, engineering documents, legal contracts, and all business-critical correspondence.

10.4.2 Business problems and their solutions

Based on the business scenario, the company has the following business problems:

- No single view of all customer data across all locations
- No consistent cost analysis or reporting on internal processes
- Inconsistent information security and retention policies
- Difficulty locating information
- Regional customization
- Reliability and scalability
- Development and support costs
- Uncontrolled file shares

No single view of all customer data across all locations

Information is fragmented and impossible to report. No single unified search for information across the organization exists.

Solution

We conducted an analysis of metadata and document classification required for customer related and Human Resources documents. The initial assessment found 60 document types with 200 metadata items shared between these classes. There are also 15 record types to consider with another 60 metadata items for these records management classes.

Future document classes will be subclassed from the above classes to enforce minimum required metadata standards. Search Templates and content indexing are set up to provide an infrastructure for finding information across all information stores and types.

These templates will also be used within business processes to find other relevant data based on initial information provided for each request, for example, a customer request for a new product can cause a business process to search for the customer's last change of address and financial details documents. All of these are presented to internal users, which provides employees with instant access to all required information to complete their tasks without the need for manual search.

No consistent cost analysis or reporting on internal processes

Executives believe that the company can achieve a great deal of cost savings by analyzing current processes and automating tasks, where possible, which includes data validation, manual system updating, and paper-oriented tasks.

Solution

Perform a business process analysis for the Customer Onboarding, Account Opening, and Customer Maintenance processes. They identify steps that can be automated and made parallel and identify what information people must adequately use to perform human tasks.

Inconsistent information security and retention policies

There is a potential for confidential information to leak out from within the company. The company is keen to use security methods from preventing any unauthorized access. There is an existing hierarchical security classification method in the paper world that they want to apply to all content. Any leak of information can result in competitive, public opinion, financial, and criminal repercussions. These risks also occur when information is kept beyond the time it is required. Legal discovery costs hit the corporation hard in previous litigations.

Solution

The company's information security hierarchy is implemented within IBM FileNet Content Manager as a Marking Set and applied to the Security Classification property of all documents within the system. Each document class sets the default for this property to the most relevant setting.

Security Policies are created to act as application domain Access Control Lists. An example of this is a Human Resources Document whose policy allows all Human Resources users to read metadata, but senior Human Resources users to view the actual content.

Certain countries have legislation that prevents employee data being sent internationally. To comply, we create a marking set called Country Visibility and populate it when needed. This action effectively denies access to any out-of-country users and is also useful when dealing with security conscious government customers of the company.

Customer interactions generate cases within IBM FileNet Advanced Case Management. All information is created to handle a customer request, and all correspondence are kept together in one case folder. After a vital business interaction is complete, such as contract signing or account opening, these documents and the managing processes are declared as critical business records to prove compliance.

Difficulty locating information

Employees find it hard to do their day-to-day activities because the information that they require is spread out between web-based systems (including wikis and blogs), collaboration tools, email, file shares, paper, and existing electronic repositories. The organization wants to replace these eventually, but first wants to make all of this content accessible to users through search and internal business processes.

Solution

Remove barriers in accessing information that is relevant to employees doing their job by providing federation at the content level to existing systems, migrating some systems that are not web or ECM interface accessible (such as file shares), and linking to web-based systems. Link this all together at the user interface level to provide all information that is required to make a decision on the same first summary window. Provide links to often used but not mandatory information, such as best practice guides, pricing rules, or Business Intelligence displays.

In addition to this process-orientated view of information, it is often necessary to search for answers across all existing repositories. Historically, search has broken down across vast, and various sources of information because it cannot correctly categorize and index such diverse information domains. By taking the metadata and classification described in the previous point, we can apply that to other information sources to make finding disparate but related data easier. This process can greatly improve enterprise search result accuracy by providing domain summary information to search by. At a simple level, this can be item types, such as news item or product information, but at a more detailed level you can aggregate information search results by, for example, country, product, or customer industry.

This solution also makes later transition to a common enterprise content management storage mechanism for all applications much easier to accomplish because all data is described using the same metadata schema.

Regional customization

Internationalization and localization are of paramount importance. The company promotes itself as an international player with local focus. As such, it tries to provide information in all languages, both national and local.

Information might be created in one world region, but primarily consumed in another. Bandwidth to some locations is limited. An advanced feature of this cache is the write-through caching capabilities. This means that if a user in New York checked in a document into the Los Angeles-based object store, this document is automatically cached in the New York content cache. This capability helps reduce the bandwidth usage and greatly reduce use of the network for often used, but rarely changed, documents.

Solution

By using the same underlying content and process metadata schemes, we can ensure that all content, regardless of origin, meets minimum indexing requirements. Many tools can be used to map country and language (locale) named data into this standard schema. An electronic form, for example, can have the same fields on it, but have a version translated from English with its left-to-right text into a right-to-left language with completely different local terminology and instructions. Other web user interfaces, such as IBM FileNet Advanced Case Management, can have language packs installed and detect the user's locale to show the interface in the most appropriate language.

Reliability and scalability

If a system goes down, it directly impacts the company in terms of revenue generation and customer satisfaction. The company is keen to maintain the highest levels of uptime and resilience. They carry out over a hundred million customer transactions per day. Every minute they are down, it costs them tens of thousands of dollars.

Solution

The content caching features of IBM FileNet Content Manager can be used to cache a document after the first authorized request at a remote location. So if a document in Los Angeles was requested by an employee working on the islands in the English Channel, for example, the first time they requested it there is a lag. The next time any authorized user requests the same document, however, it is drawn from the local cache.

As we saw in previous solutions, we can provide horizontal scaling to meet high availability requirements. When a full site goes down, or is cut off from the network, another approach is required. Disaster recovery can be provided by having a duplicate server infrastructure for a site, either active or passive, at another location. When a disaster happens in the primary site, the duplicate server is made available to clients by making a network routing change and pointing all other sites to this duplicate site.

Data integrity is maintained by providing background, behind the scenes, replication of file storage and database tables that the first site uses. This implementation means that even if an entire site is unavailable for a full day because of network issues beyond the company's control, the alternative site is made available within minutes, enabling the rest of the organization to work on this information and processes as normal, mitigating business risks due to down time.

Development and support costs

There are many existing systems that the company wants to eventually replace with off the shelf software wherever possible. As a stop gap method, however, they want to quickly replace the storage layer of their systems with an ECM persistence layer. They want to also ensure that in the future, any new systems can use the same storage layer technology. Because the company is risk adverse, they want to ensure that this layer completely abstracts the underlying content repository, which shields them from any future vendor-specific API changes or changes of vendor.

Solution

Large organizations are increasingly looking at a service-oriented architecture approach to mitigate future proofing issues to do with software upgrades, dependencies, and migrations. As such, companies created a shared service business layer that abstracts content creation and retrieval to provide a managed shared service for their organization.

This implementation results in a single, independent interface using open standards to access content. It also makes the back end architecture transparent to the application. This is practical when you must monitor and charge for storage and access to other internal teams. It also makes migrating back end storage of content much easier to administer because you only need to update the mappings to storage on the shared service business tier, not in each and every application that might use the content, which you can achieve by using either a layer or an enterprise content management system that supports SOA and federation, such as the IBM FileNet Content Manager. You can use the IBM FileNet P8 stack with its built-in Content Federation Services (CFS) mappings to other content management and heritage systems. The additional advantage, other than the maintenance cost advantage, of using CFS is that the add-on products all then have visibility over the content. This visibility is particularly useful for adding process accessible content to your application that is built on top of IBM FileNet Business Process Manager and IBM FileNet Advanced Case

Management. In this scenario, federation, as opposed to a intermediary service API, can achieve greater performance and usage features.

IBM FileNet Content Manager and IBM FileNet Business Process Manager are fully accessible, which we learned in this book, through Web services and other APIs. You can even invoke and interact with individual running processes using Web services. This fits perfectly into a service-oriented architecture, as required by the Shared Service or Software as a Service (SaaS) models.

Uncontrolled file shares

There will be a migration from existing departmental and user file shares into the new system to remove duplicates and reduce storage capacity. There are currently billions of documents over 50 file servers. Bulk classification is error prone; whereas, manual classification is too costly to perform. The company cannot find a good solution to this.

Solution

Content Collector for File System migrates the more well-defined departmental file shares. Rules are developed to classify documents of particular types, names, and filing locations into specific classes. The unstructured user shares, however, require a more flexible and automated approach. IBM Classification Module can be used with Content Collector to suggest the most likely class and filing location for the documents on the share.

As we mentioned in 6.3, “IBM Classification Module” on page 144, IBM Classification Module works by teaching its neural networks with a corpus of documents of a specific classification and filing location. The larger the corpus you train with, the higher the accuracy. Any documents that the system is unsure about will be routed for manual processing. In this situation, the user sees the suggested classes and filing locations and either agree to them or choose an alternative. As IBM Classification Module learns to be more accurate over time, the accuracy score will improve, causing less and less documents to pass through the manual process.

You might choose, for example, to get 100 internal employees to ingest 100 of their documents and provide the correct classification and filing locations. This process results in a training corpus of 10000 documents. You can then use this as a basis for the automated migration, perhaps migrating a portion of your users’ content per week.

After the migration is complete, the Classification Module system can continue to be used to match new incoming documents. At this stage, it is well trained, and can be used to help users classify new documents, or by the system for incoming emails or OCR text from scanned images. This process greatly reduces user error, increases employee buy-in for using the system, and prevents users from sticking to what they are used to, that is continuing to use personal storage and file shares for new documents.

10.4.3 Customer architectural constraints

The preferred large-scale system is IBM p-Series with AIX and LPARs to support virtualization. Other preferred internal systems include Tivoli Directory Server, Tivoli Access Manager with WebSeal for SSO, WebSphere Application Server, DB2 RDBMS.

Major regional centers include Los Angeles, London, Dubai, Hong Kong, and Johannesburg. Offshore customers are handled out of New York and the Channel Islands (English Channel, UK). This setup connects to the London regional hub. Several smaller offices throughout each region, Los Angeles, London, and Hong Kong, each handle approximately 10000 employees, with Johannesburg and Dubai taking approximately 5000 each in their regions.

The company has two million customers worldwide ranging from individuals up to large multi-nationals. On average, one million customers sign up for new products and services every year with each request requiring an average of 10 documents of 40KB being supplied by the customer for the onboarding process. 80 percent of these documents are eForms. A further 20 documents are generated internally (10 eForms approvals, 10 documents to be sent out) on average, for each request before completion. Individuals account for 95 percent (950,000) of the company’s new requests with the remainder (five percent, 50,000) being from companies.

The company has 40,000 employees. Internally the company has a team of 100 Human Resources professionals (25 senior employees). Focus has 2,000 people managers, and 10,000 employees working on customer facing duties.

The client onboarding process consists of 10 independent business processes with an average of 15 human steps. Of these, five are automated in the to be process, two are data validation, two are system queries, and one is a system update. There will be an additional 10-system steps to manage content. It currently takes 30 days to complete a product onboarding request.

There are Human Resources Onboarding, Employee Leaves, Change of Details, Change of Manager, Change of Role, Employee Review, Disciplinary Action and Document Access Request processes. Typically, there is a 10 percent attrition rate among employees. Human Resource Onboarding process requires that 10 documents are imported or generated (five of these are eForms). Employee Review and Disciplinary Action processes generate five documents (all are eForms). On average, these processes have five human steps. Of these five steps,two are information validation and discovery steps.

The document access process is currently manual and paper based and is used 60,000 times a year. It also takes two weeks to complete. These both need reducing by setting default document security and implementing new streamlined employee leaving, change of role, and change of manager processes. The target processing time is five days.

10.4.4 Solution architecture

Disaster recovery can be achieved by mirroring each site at a nearby, but independent, facility. Set up disaster recovery sites as hot standby that normally do not receive user requests. High availability is designed into each site's infrastructure.

We call Los Angeles, London, and Hong Kong high-load sites, and Dubai and Johannesburg medium-load sites.

For load modelling, we assume that all client onboarding customers are spread throughout each region, according to the number of employees at each regional office (the regional staffing is directly proportional to the number of customers in that region). These requests are handled on the new follow-the-sun operating model.

All Human Resources requests are handled in region and do not follow the follow-the-sun model, which are spread out as per the number of employees per region.

Figure 10-4 on page 358 shows the large-site system architecture. We use mainly IBM p-Series machines. The number of instances and assigned CPUs are based on sizing from IBM SCOUT (which we do not cover in this book). Although not shown, the system also needs load balancing for Workplace/Workplace XT, Content Engine, and Process Engine. We also have not included IBM Classification Module, Content Collector or Content Federation Services (CFS) in the solution diagram. In our scenario, we only talk about using these for migrating content.

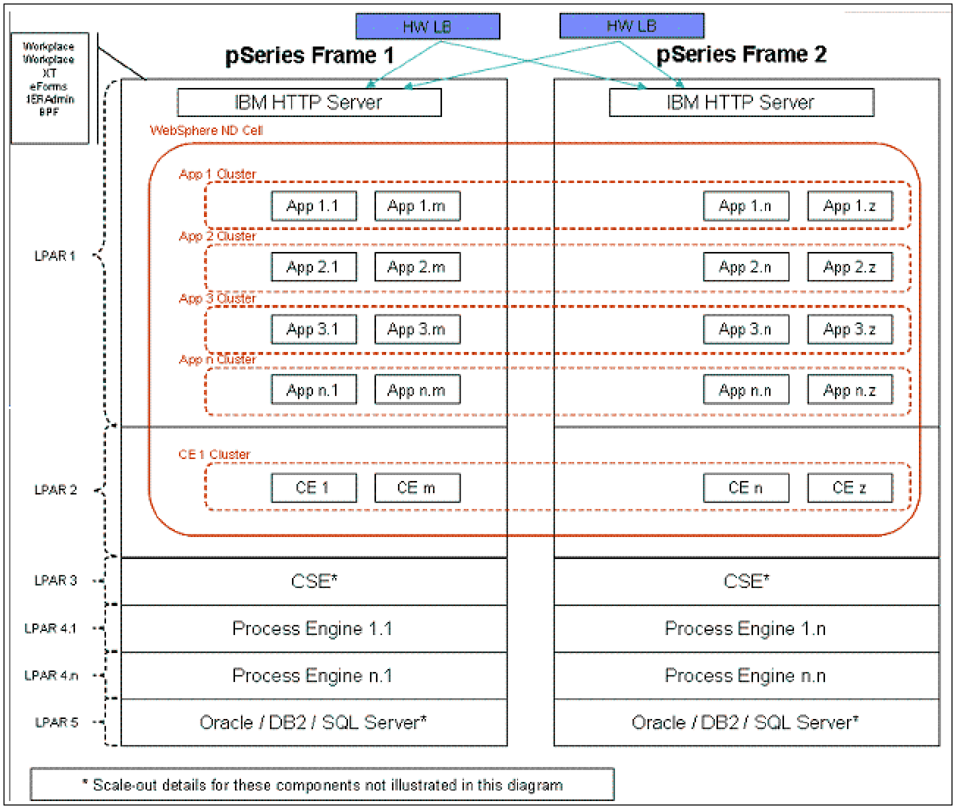


Figure 10-4 Large site system architecture

In Figure 10-4, we only show servers from a sunny-day scenario, which means that we assume that no servers ever fail; therefore, we have not installed a highly-available service. In practice, thanks to clustering technologies, making a service highly available is easily facilitated by adding extra load handling nodes and has an automatic failover mechanism that is transparent to the client. Having a network layer with virtual hosts for the clustering and farming mechanism makes this transparent to the client application.

A

User accounts

This appendix contains a list of accounts, their function, and the minimum permission requirements concerning the FileNet P8 environment for installation, maintainance, and application administration.

Security requirements of user accounts

This table refers to and enhances “Security requirements for installing IBM FileNet P8” on page 212 functions and security permissions.

Note: All account names in the table are placeholders, except they are mentioned explicit as mandatory or recommended names. As long the user accounts are not defined as “technical user accounts”, the account names of actual administrator accounts should be used for group memberships and installation purpose.

Table 10-2 Security requirements of user accounts

Function	Security permission (minimum requirement)	Account name and remarks
Content Engine service user (LDAP read access)	**Active Directory** *Read access* rights (specifically the *Read All Properties permission*) to the forest-wide configuration directory partition and the domain directory partition in each desired domain in the Active Directory forest.	*ce_service_user* Technical user account for permanent configuration. The directory service user cannot be accessed using referrals. Because *Authenticated Users* by default is a member of the *Pre-Windows 2000 Compatible Access group* which has these permissions, it will needed to assign the permissions to *ce_service_user* only if the default is modified or *Authenticated Users access rights* are restricted.
	AD LDS The account *ce_service_user* must be member of CN=Readers of the the CN=Roles container in the partition (not under the CN=Configuration.)	*ce_service_user* The directory service user cannot be accessed using referrals.
	Sun Java System Directory The rights Read, Search, Compare have to be granted to the *ce_service_user*	*ce_service_user* The directory service user cannot be accessed using referrals.
	Novell eDirectory The rights Read, Compare have to be granted to the *ce_service_user.*	*ce_service_user* The directory service user cannot be accessed using referrals.
	CA Directory The rights Read, Search, Compare have to be granted to the *ce_service_user.*	*ce_service_user* The directory service user cannot be accessed using referrals.
	IBM Tivoli The rights Read, Search, Compare have to be granted to the *ce_service_user.*	*ce_service_user* The directory service user cannot be accessed using referrals.
	Oracle Internet Directory The rights Read, Search, Compare have to be granted to the *ce_service_user.*	*ce_service_user* The directory service user cannot be accessed using referrals.

Function	Security permission (minimum requirement)	Account name and remarks
Content Engine 3.5 upgrade user	► Read access to the Content Engine 3.5 directory (default location is C:\Program Files\IBM\FileNet\Content Engine). ► Read access to the 3.5 version of sysinit.dat. ► Read/write access to the root directories for the file storage areas ► Full Control access to the FileNet P8 domain object (in other words, a gcd_admin) ► Full Control access to all object stores in the FileNet P8 domain (in other words, an object_store_admin for all object stores in the FileNet P8 domain)	*ce_upgrade_user*
Content Engine system user (bootstrap administrator)	The account must be a directory server account that has been configured as an application server administrator.	*ce_bootstrap_admin* Technical user account for permanent configuration. A directory service and application server administrative account that is stored in the CEMPBoot.properties file that is archived in the Content Engine EAR file.
G CD Administrator	To be grant ed *Full Control access* to the Content Engine domain object.	*gcd_admin* or *gcd_admin_group* A directory service account that has *Full Control access* to the Content Engine domain object. Object store administrative rights do not include the ability to add, move, or remove object stores, fixed content devices, content cache areas, or any of the other FileNet P8 domain resources. These permissions are granted only to GCD administrators.

Function	Security permission (minimum requirement)	Account name and remarks
Object Sore Administrators	To be granted *Full Control access* to one or more object stores to an *object_store_admin* or *object_store_admin_group*	*object_store_admin* or *object_store_admin_group* By default, the *GCD administrator* creating the object store also becomes an object store administrator, but this account can be removed if the security design requires dedicated accounts for each object store and GCD.
K2 security group	**Windows** *k2_sec_group* must be an Active Directory group when IBM Legacy Content Search Engine is installed on a machine in a Windows domain. *k2_sec_group* must be a Windows local group when IBM Legacy Content Search Engine is installed on a machine in a Windows workgroup. **UNIX** *k2_sec_group* must be a UNIX operating system group.	*k2_sec_group* Enterprise Manager automatically places the value assigned to the *k2_sec_group* on each Verity Collection. K2 security user accounts must be members of the *k2_sec_group*. Only members of this group will have access to the collection.
K2 security user	Must be defined as an authorized K2 administrator in the K2 dashboard. This user must be a member of the k2_sec_group. Only members of this group will have access to the collection.	*k2_sec_user* Technical user account for permanent configuration.
Process Engine service user	This user must belong to the Process Engine Administrator group.	*pe_service_user* Process Engine uses the pe_service_user when connecting to the Content Engine server.
Process Engine administrator group	Members of this group automatically have administrative privileges for Process Engine.	*pe_admin_group* A directory server group whose members can use Process Task Manager to manage Process Engine
Process Engine configuration group	If this group is used to configure security on Process Task Manager, members of this group or of the Process Engine Administrator Group (*pe_admin_group*) can make configuration changes to the workflow database. If the Process Engine Configuration group is not used during this configuration, anyone can make these changes.	pe_config_group (Optional) Members of this group automatically have configuration privileges for the Process Engine workflow database.

Function	Security permission (minimum requirement)	Account name and remarks
Process Engine region administrator	Full Control access rights on the FileNet P8 domain. Membership in the Process Engine pe_admin_group	*pe_region_admin* A directory server user account that has Full Control access rights to the FileNet P8 domain. The *pe_region_admin* therefore has permissions equivalent to the *gcd_admin* but should be used only for Process Engine purposes, The *pe_region_admin* credentials are entered in Process Task Manager and in PEInit, to create or modify Process Engine isolated regions
Content Engine database accounts	MS SQL Server The *ce_db_user* has to be granted at least the following database access permissions: ▸ db_datawriter ▸ db_datareader ▸ db_ddladmin ▸ public All ce_db_user accounts to SQL Server's master database and grant the public role to each. When configuring the procedure "Configuring the JDBC distributed transaction components", these accounts will also be granted the SqlJDBCXAUser role. The *ce_db_user* must be a SQL Server account. It does not have to be an account in the configured directory service.	*ce_db_user* Technical user account for permanent configuration. Either a single user account for database access will be introduced or different user accounts for different database purposes Example: One user (e.g. *ce_db_user1*) for the GCD database and different accounts for each object store (e.g. *ce_db_user2* , *ce_db_user3*, etc.).

Function	Security permission (minimum requirement)	Account name and remarks
Content Engine database accounts	Oracle database Each *ce_db_user* has to be granted least the following permissions: ► CREATE SESSION ► ALTER SESSION ► CREATE TABLE ► CREATE VIEW ► CREATE SEQUENCE (Object Store creation only) ► Alter user set QUOTA UNLIMITED on all tablespaces used by db user ► SELECT on pending_trans$ ► SELECT on dba_2pc_pending ► SELECT on dba_pending_transactions ► SELECT on DUAL ► SELECT on product_component_version ► SELECT on USER_INDEXES (Upgrade Content Engine only) In addition: ► On Oracle 10.2.0.4, Oracle 11.1.0.6.0 or later (where both the JDBC client and the database server are at those levels), grant EXECUTE on dbms_xa ► For Oracle releases earlier than 10.2.0.4, see Oracle Metalink document ID 436362.1 for required patches, or grant this instead EXECUTE on dbms_system Several of these permissions are required by Content Engine JDBC XA transactions.	*ce_db_user* Technical user account for permanent configuration. One account for each object store tablespace and one for the GCD tablespace shall be used.

Function	Security permission (minimum requirement)	Account name and remarks
Content Engine database accounts	**Content Engine DB2 for Linux, UNIX and Windows accounts** The following database permissions have to be granted: ▶ CONNECT ON DATABASE ▶ CREATETAB ▶ USE OF TABLESPACES ▶ SELECT on SYSIBM.SYSVERSIONS ▶ SELECT on SYSCAT.DATATYPES (Object Store creation only) ▶ SELECT on SYSCAT.SYSINDEXES, SYSIBM.SYSDUMMY1 (Upgrade only) ▶ USAGE on workload SYSDEFAULTUSERWORKLOAD ▶ IMPLICIT_SCHEMA on DATABASE For added security in a shared database environment, you can remove the Connect privilege from the Public group	*ce_db_user* Technical user account for permanent configuration. Separate accounts can be used for each object store, but are not required.
	Content Engine database user (DB2 for z/OS) The following database permissions have to be granted: ▶ GRANT SYSADM TO ce_db_user ▶ GRANT USE OF STOGROUP storagegroupname TO ce_db_user ▶ GRANT USE OF BUFFERPOOL buffer_pool_name TO ce_db_user ▶ GRANT SELECT ON SYSCAT.DATATYPES TO ce_db_user ▶ GRANT SELECT ON SYSIBM.SYSVERSIONS TO ce_db_user	*ce_db_user* is the alias for the operating system user name created for the Content Engine database user *storagegroupname* is the alias for the name of the storage group *buffer_pool_name* is the alias for the name of the buffer pool (Note: DB2 for z/OS does not allow underscores in account names) Separate accounts can be used for each object store, but are not required.
Process Engine database accounts	**Process Engine database user for MS SQL Server** he *pe_db_user* has to be granted at least the following database access permissions: ▶ db_datawriter ▶ db_datareader ▶ db_ddladmin ▶ public	*pe_db_user* In a farmed or cluster configuration, each Process Engine must be configured to use the same database user name. Minimum required permissions

Function	Security permission (minimum requirement)	Account name and remarks
	Process Engine database user for Oracle databases Oracle database Each *ce_db_user* has to be granted least the following permissions: ► CREATE SESSION ► ALTER SESSION ► CREATE TABLE ► CREATE VIEW ► Alter user set QUOTA UNLIMITED on all tablespaces used by db user ► SELECT on pending_trans$ ► SELECT on dba_2pc_pending ► SELECT on dba_pending_transactions ► SELECT on DUAL ► SELECT on product_component_version In addition: ► On Oracle 10.2.0.4, Oracle 11.1.0.6.0 or later (where both the JDBC client and the database server are at those levels), grant EXECUTE on dbms_xa ► For Oracle releases earlier than 10.2.0.4, see Oracle Metalink document ID 436362.1 for required patches, or grant this instead EXECUTE on dbms_system	*pe_db_user* In a farmed or cluster configuration, each Process Engine must be configured to use the same database user name.

Function	Security permission (minimum requirement)	Account name and remarks
	Process Engine database user for DB2 for Linux, UNIX and Windows The following database permissions have to be granted: ▶ CONNECT ON DATABASE ▶ CREATETAB ▶ USE OF TABLESPACES ▶ SELECT on SYSIBM.SYSVERSIONS ▶ SELECT on SYSCAT.DATATYPES (Object Store creation only) ▶ SELECT on SYSCAT.SYSINDEXES, SYSIBM.SYSDUMMY1 (Upgrade only) ▶ USAGE on workload SYSDEFAULTUSERWORKLOAD ▶ IMPLICIT_SCHEMA on DATABASE For added security in a shared database environment, the Connect privilege can be removed from the Public group	*pe_db_user* In a farmed or cluster configuration, each Process Engine must be configured to use the same database user name. Databases with the RESTRICTIVE option are not supported.
	Process Engine database user for DB2 for z/OS ▶ GRANT DBADM ON DATABASE *databasename* TO *operatingsystemuser* ▶ GRANT USE OF STOGROUP *storagegroupname* TO *operatingsystemuser* ▶ GRANT USE OF BUFFERPOOL *buffer_pool_name* TO *operatingsystemuser* ▶ GRANT SELECT ON SYSCAT.DATATYPES ▶ GRANT SELECT ON SYSIBM.SYSVERSIONS	*operatingsystemuser (pe_db_user)* is the alias for the operating system user name on the database server created for the Process Engine database user *storagegroupname* is the alias for the name of the storage group *buffer_pool_name* is the alias for the name of the buffer pool (Note: DB2 for z/OS does not allow underscores in account names) Separate accounts can be used for each object store, but are not required.

Function	Security permission (minimum requirement)	Account name and remarks
(Web-) Application server	**Content Engine application server administrator and administrative console user** **IBM WebSphere Application Server (WAS)** ▸ This administrative account can be the same as the Content Engine system user (*ce_bootstrap_admin*). If the same credentials for both in Configuration Manager are entered, and provided that an LDAP account is used, and also that the account will be a *P8 Domain Administrator* (*gcd_admin*) specified during GCD creation. ▸ If the site uses Federated Repository, *ce_appserver_console_admin* must be a unique user across all federated realms including the WAS local repository. ▸ After Content Engine installation is complete, the account's permissions can be reduced to a lesser role within WAS, such as *Configurator.* The user account *ce_appserver_admin* must have full administrative control over the WAS domain that contains the Content Engine. **Oracle WebLogic Application Server** *ce_appserver_admin* and *ce_appserver_console_admin* must be different accounts **JBoss Application Server** does not require an administrative account.	*ce_appserver_admin* (can also be used as *ce_install_user*) and *ce_appserver_console_admin, ce_appserver_install_group* The *ce_appserver_admin* account is used to perform the following tasks: ▸ Create directory service providers within the application server. ▸ Create JDBC providers and data sources for database connectivity. ▸ Deploy the Content Engine application. ▸ Stop and restart servers and cluster members (WAS). ▸ Stop and restart managed servers (Oracle WebLogic Server). ▸ When you enable WAS Global Security as a post-install step, you must use *ce_appserver_console_admin* to login to the WAS console and run *Configuration Manager* and any scripts.

Function	Security permission (minimum requirement)	Account name and remarks
Installation account on the Workplace server / WP XT installation	**Workplace or Workplace XT installer, deploy & administrative account (Windows)** This account must be a MS Windows local administrator or a user with equivalent permissions. The installer account (*wp_install_user* or *wpxt_install_user*) must be granted read/write/execute permission to these directories and files: Installation paths: **WebSphere Application Server** WAS_HOME/profiles/default/installedApps/node_name/app_engine_war.ear/app_engine.war *WAS_HOME/profiles/default/config/cells/machine_name/Node01cell/nodes/machine_name/Node01/serverindex.xml* **WebLogic 10.x** *BEA_Home/bea/wlserver_10.0/server/bin/startWLS.sh or start WLS.cmd* *BEA_home/bea/user_projects/domains/domain_name/config/config.xml* **JBoss** *JBOSS_home/bin/run.sh or run.bat* *JBOSS_home/server/default/conf/login-config.xml* *(on both Content Engine and Workplace servers)*	*wp_install_user* or *wpxt_install_user* as well as *wp_deploy_user* or *wpxt_deploy_user* as well as *wp_admin_user* or *wpxt_admin_user* Recommendation: In order to enable the same persons to install and deploy the FileNet software the permissions should be configured by groups and the installer indirect by relating to the appropriate groups.

Function	Security permission (minimum requirement)	Account name and remarks
	Workplace or Workplace XT installer, deploy & admin (UNIX) The installer account (*wp_install_user* or *wpxt_install_user*) must be granted read/write/execute permission to these directories and files: Installation paths **WebSphere Application Server** *WAS_HOME/profiles/default/installedApps/node_name/app_engine_war.ear/app_engine.war* *WAS_HOME/profiles/default/config/cells/machine_name/Node01cell/nodes/machine_name/Node01/serverindex.xml* **WebLogic 10.x** *BEA_Home/bea/wlserver_10.0/server/bin/startWLS.sh or start WLS.cmd* *BEA_home/bea/user_projects/domains/domain_name/config/config.xml* **JBoss** *JBOSS_home/bin/run.sh or run.bat* *JBOSS_home/server/default/conf/login-config.xml (on both Content Engine and Workplace servers)*	*wp_install_user* or *wpxt_install_user* as well as *wp_deploy_user* or *wpxt_deploy_user* as well as *wp_admin_user* or *wpxt_admin_user* Recommendation: In order to enable the same persons to install and deploy the FileNet software the permissions should be configured by groups and the installer indirect by relating to the appropriate groups.

Function	Security permission (minimum requirement)	Account name and remarks
Content Engine application server installation	The following accounts must be member of the *ce_appserver_install_group*: ▸ *ce_appserver_install_user* ▸ *ce_install_user* ▸ *config_mgr_user*	ce_appserver_install_group and *ce_appserver_install_user* The user accounts in ce_appserver_install_group will perform the following tasks: ▸ Give operating system privileges to the directories used for Content Engine installation and for the application server's instance/domain/profile. ▸ Configure and deploy the Content Engine EAR files which require access to the application server's instance/domain/profile directories. ▸ Have permissions on devices/drives to read and write that are designated for external Content Engine file storage.
Content Engine operating system user	**Content Engine operating system user account** Windows ▸ For Content Engine and file storage areas, *ce_os_user* must reside in the same Windows domain or in trusted Windows domains as the servers that host Content Engine and the file storage area. ▸ For file storage areas and using WebSphere the WebSphere service must be set to logon as the *ce_os_user*. UNIX ▸ For Content Engine and file storage areas, configuring security requires the use of NFS.	*ce_os_user* An operating system account that must be used to log on as to create and configure the shared root directory of a file storage area or content cache area. From a practical standpoint, the account that is used to install the application server should be the same account that is used to start the application server process. As an administrator, one will always log in using the same *ce_os_user* account to secure the folders and files in the file system that Content Engine will use for a file storage area.

Function	Security permission (minimum requirement)	Account name and remarks
Content Engine configuration manager	**Content Engine configuration manager user** *config_mgr_user* must belong to the *ce_appserver_install_group*. (Windows only) For using Active Directory tools, the *config_mgr_user* must be added to either the *Power Users* group or the *local administrators* group. At several points in the installation process one will be instructed to grant additional permissions to *config_mgr_user,* including the following permissions: ► Execute permission to the Configuration Manager executable file, *configmgr.exe* (Windows) or *configmgr.sh* (UNIX). ► Read and write permission to the directory where Configuration Manager will create the configuration XML files. For example: – the directory you specify using the optional -path parameter when you run Configuration Manager. – the default directory, *ce_install_path/tools/configurationmanager/tasks*, if you do not specify a path parameter.	*config_mgr_user* operating system account used to run Configuration Manager.

Function	Security permission (minimum requirement)	Account name and remarks
Process Engine installer account	**Process Engine installer account** **Windows installations** ► Read, write, and execute permissions to the device or location where Process Engine is to be installed, by default *C:\Program Files\IBM\FileNet\ProcessEngine.* ► Read, write, and execute permissions to the device or location where Process Engine common files are to be installed, by default *C:\Program Files\IBM\FileNet\Common Files* ► Write permission on the *.\Common Files\taskmaninstances.xml* file, if the *P8TASKMAN_HOME* environment variable exists. ► Write permission on the *%tmp%* directory If the installation has been executed as a non-administrative user, one must log on after the installation as an administrator to run a script to edit the registry and create the *Process Engine Manager service*. **UNIX installations** ► Read, write, and execute permissions to the device or location where Process Engine is to be installed, by default */opt/IBM/FileNet/ProcessEngine.* ► Read, write, and execute permissions to the device or location where Process Engine common files are to be installed, by default */opt/IBM/FileNet/CommonFiles*. ► Write permission on the *./CommonFiles/taskmaninstances.xml* file, if the *P8TASKMAN_HOME* environment variable exists. ► Write permission on the *%tmp%* directory If the Process Engine shall be to configured to start automatically, this user must also have root permission to modify the */etc/inittab* file.	pe_install_user Operating system user account to log on as to run the Process Engine installation program

Function	Security permission (minimum requirement)	Account name and remarks
K2 operating system user	**IBM Legacy Content Search Engine operating system user** If IBM Legacy Content Search Engine is to be in a Windows workgroup, *k2_os_user* must be given the Log on as a service right in the Local Security Policy. If IBM Legacy Content Search Engine is to be in a Windows domain, *k2_os_user* must be given the Log on as a service right in the Domain Security Policy. UNIX processes will run as this user. *k2_os_user* can be an unprivileged user; however, if the system is using SHADOW PASSWORDS then the *vspget* process must run as the root user. Windows-based file storage areas: *k2_os_user* must have Read access to the file storage area root directory and to the file storage area's UNC path. Unix-based file storage areas: one way to grant access to the *k2_os_user* is to set the Set UID bit of all the K2 program files, and then also set the owner of each program file to the *k2_os_user*, which must also have Read/Execute access to the file storage area root directory and to the file storage area's NFS path. *k2_os_user* requires Read and Write permissions to the K2 collections directory and the collections temp directory. The IBM Legacy Content Search Engine software needs to read the file storage areas and write the full text index collections as part of the full text indexing operation.	*k2_os_user* Windows services will run as *k2_os_user.* This user account must be given administrator privileges for Windows on all machines on which the IBM Legacy Content Search Engine software will be installed.
CSS operating system user	I**BM Content Search Services operating system account** This account must be an operating system user with rights to run the *IBM Content Search Services* startup and shutdown commands. By default, the *css_install_user* can also run these commands.	*css_os_user*
CSS installer user	**IBM Content Search Services installer account** On Windows, this account must be a Windows Local administrator or a user with equivalent permissions. Read/write/execute permission to the *css_install_path.*	*css_install_user*

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- *IBM FileNet Content Manager Implementation Best Practices and Recommendations,* SG24-7547
- *Introducing IBM FileNet Business Process Manager*, SG24-7509
- *Understanding IBM FileNet Records Manager*, SG24-7623 (IBM FileNet Records Manager is currently known as IBM Enterprise Records)
- *IBM High Availability Solution for IBM FileNet P8*, SG24-7700
- *Disaster Recovery and Backup Solutions for IBM FileNet P8 Version 4.5.1 Systems*, SG24-7744
- *Federated Content Management: Accessing Content from Disparate Repositories with IBM Content Federation Services and IBM Content Integrator*, SG24-7742

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks publications, at this Web site:

ibm.com/redbooks

Online resources

These Web sites are also relevant as further information sources:

- IBM FileNet P8 Platform main information page

  http://www.ibm.com/software/data/content-management/filenet-p8-platform

- IBM FileNet P8 Version 5.0 Information Center

  http://publib.boulder.ibm.com/infocenter/p8docs/v5r0m0/index.jsp

- IBM FileNet P8 Platform product documentation

  http://www.ibm.com/support/docview.wss?rs=3247&uid=swg27010422

  The above URL includes links to all expansion IBM FileNet P8 products.

- IBM FileNet Content Manager

  http://www.ibm.com/software/data/content-management/filenet-content-manager

- IBM FileNet Business Process Manager

  http://www.ibm.com/software/data/content-management/filenet-business-process-manager

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

IBM

Redbooks

IBM FileNet P8 Platform and Architecture

(0.5” spine)
0.475”<->0.875”
250 <-> 459 pages

IBM FileNet P8 Platform and Architecture

Architecture and expansion products

Solution design, creation, and implementation

Security, infrastructure, and scalability information

IBM FileNet P8 Platform is a next-generation, unified enterprise foundation for the integrated IBM FileNet P8 products. It combines the enterprise content management with comprehensive business process management and compliance capabilities. IBM FileNet P8 addresses the most demanding compliance, content, and process management needs for your entire organization. It is a key element in creating an agile, adaptable enterprise content management (ECM) environment necessary to support a dynamic organization that must respond quickly to change.

In this IBM Redbooks publication, we provide an overview of IBM FileNet P8 and describe the core component architecture. We also introduce major expansion products that extend IBM FileNet P8 functionality in the areas of content ingestion, content accessing through connectors and federation, the application framework, and discovery and compliance. In this book, we discuss the anatomy of an ECM infrastructure, content event processing, content life cycle, and business processes.

This book gives IT architects, IT specialists, and IT Technical Sales a solid understanding of IBM FileNet P8 Platform, its architecture, its functions and extensibility, and its unlimited capabilities.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-7667-01 ISBN 0738435317